撮像系の物理モデルに基づく動画像からの３次元復元手法の高精度化に関する研究

博士学位論文

撮像系の物理モデルに基づく動画像からの３次元復元手法の高精度化に関する研究

平成28年度

(1月10日 提出)

東北大学大学院情報科学研究科

システム情報科学専攻

伊藤　栄介

Towards Accurate Three-dimensional Reconstruction from Video Imagery Based on Physics-based Modeling of Imaging Systems

Eisuke Ito

Abstract

The up-to-date development of computer technology has enabled wider use of three-dimensional computer graphics (3D CG) in creating chinemas and computer games. While in the past 3D CGs were manually created by creators from scratch, methods of measuring images and depth information with sensors and estimating shapes of people and objects have become used widely in recent years. Similar techniques are also applied to various problems in other fields such as terrain surveying and pose estimation of mobile robots. In addition to such applications for high-end users, recently, there are increasing demands for consumer-oriented applications, as good-quality images can be easily acquired with smartphones. An example is augmented reality (AR) that superimposes the image of a virtual object created by CG on real images. It is expected that display of additional information on the image of a real scene improves the usability and widening of playing range etc. In such AR applications, it is necessary to calculate the relative pose between the camera and the target object from its acquired images. Although in some of these applications, depth sensors are mainly used to obtain geometric information, it is possible to compute camera poses and shapes of target objects only from images captured by a camera. In the field of computer vision, this problem is referred to as structure from motion (SfM), which has been studied for many years as a major research subject in the field.

The fundamental theory of SfM has been well established owing to a large amount of previous studies, some of which has also been used in practice. However, SfM has not been fully used to solve problems to which it can potentially be applied. There are many cases where computational pipeline of SfM fails due to lack of proper modeling of imaging systems. Thus, existing methods and systems can work properly in a laboratory environment but often cannot in real-world environments. Solving this problem is indispensable for expanding the application range of SfM.

Methods for SfM can be roughly classified into two categories. One is the feature-based methods, which first extract features such as points from images and then use them to compute geometry. The other is the direct methods, which solve the optimization problem 'directly' using the images in particular image brigntness of each individual pixel. As the advantages and disadvantages of these two classes of methods are complementary, there are appropriate problem settings for each. In this research, we aim to develop methods to deal with various problems that are obstacles to expand the application range of SfM by using the advantages of the feature-based methods and direct methods separately.

In this thesis, we first focus on the problem of rolling shutter (RS) in feature point based SfM. Employing an RS camera model with linearized pure rotation, we show that the RS distortion can be approximately expressed by two internal parameters of an "imaginary" camera plus one-parameter nonlinear transformation similar to lens distortion. We then reformulate the problem as self-calibration of the imaginary camera, in which its skew and aspect ratio are unknown and varying in the image sequence. In the formulation, we derive a general representation of CMSs. We also show that our method can explain the CMS that was recently reported in the literature, and then present a new remedy to deal with the degeneracy. Our theoretical results agree well with experimental results; it explains degeneracies observed when we employ naive bundle adjustment, and how they are resolved by our method.

We next consider a direct method for tracking a planar surface in the scene, which can be used as a building-block of SfM. We show that effective degradation of image resolution will deteriorate the performance of tracking algorithms. This effective degradation of image resolution means that, when the size of the image region corresponding to the target planar surface becomes small in some cases, such as the plane moves to a distant location or the orientation of the plane has a very oblique angle toward the direction of the camera, the resolution of the obtained image will be seriously degraded. In these cases, the tracking performance deteriorates for feature-based methods as well as for appearance-based methods. Already, it is pointed out that there are several factors making tracking accuracy reduced, such as illumination changes and motion blurs for them, some effective solutions have been proposed. However the resolution degradation that we point out here has not been well recognized. In order to deal with this problem, we first model the process of image formation, specifically the sampling and the reconstruction processes in the imaging system of cameras. We then discuss how the resolution degradation affects the tracking performance. Then, we present a method that can minimize the influence of the resolution degradation. To be specific, employing a similar strategy to the

methods to deal with motion blurs, we propose to modify the template being tracked depending on the pose of the target plane during tracking. We introduce a few approximations to this model of the image formation process and the modification of the template in order to minimize the resulting increase in computational complexity. We show the effectiveness of the proposed approach through several experiments.

Finally, we consider a direct method that tracks points and lines. In recent years, the direct methods has made remarkable progress, which include those using various geometric entities such as point, single/multiple planes. In particular, the point model was able to overcome some of the difficulties with the direct methods based on planes, which has made it possible to expand the range of SfM application. However, in the case where only two images are given, it was impossible to simultaneously determine camera motion and object shapes in this approach. To overcome this problem, we extend the existing approach that tracks scene points and propose a new method that solves the problem by optimizing a cost function with respect to straight lines along with points. Considering application to general SfM, we propose a feasible optimization-based method in which points and linear elements are simultaneously treated. We show that spatial position estimation of linear part can be estimated simultaneously in our proposed method.

In summary, this paper proposes three methods. That is, the principle and method for correcting rolling shutter distortion, a method for overcoming the adverse effect of resolution reduction caused by various reasons in planar tracking, and a new direct method using points and straight lines at the same time. Through several experiments it is concluded that all these methods are effective and as a result lead to further enhancement of practicality of SfM.

目 次

第1章　序章

1.1　背景

計算機技術の発展と共に，映画やゲームの世界で 3DCG（三次元コンピュータグラフィックス）を使ったものが増えてきている．これら 3DCG は古くは人手で地道に作られてきたが，近年では画像や深度情報をセンサーで計測し，人物や物体の形状を推定する方法 [46, 60, 35] も広く使われるようになってきた．また，Google Map の地図情報を作る際の基礎になっている地形測量やロボットの自己位置推定 [14, 11] といった諸問題にも同様の技術 (Fig.1.2) が応用されている．これらの例は主にハイエンドユーザー向けの内容であるが，最近ではスマートフォンの普及によって，より手軽にカメラ画像を取得できるようになってきたため，一般向けの需要も高まってきている．応用の一例としては，Fig.1.1 のように画像上に CG などの仮想物体を重畳する AR（拡張現実感）[36, 34] が挙げられ，付加的な情報を画像上に追加することで，ユーザビリティの向上や遊びの幅が広がるなど様々な期待がされている．画像情報を使った AR では，得られた画像からカメラと対象物体との相対的な位置関係はどうなっているかを計算する必要があり，これは多数の視点から得られた画像から，対象物体の形やカメラの回転パラメータと並進パラメータを推定することによって CG の位置と向きを決定している．

以上で述べてきた様々な例では，幾何学的な情報を得るためにカメラ画像や深度センサーが主に使われているが，特に画像のみを使ってカメラ位置と姿勢，対象物体の形状を推定することをコンピュータビジョンの分野では，SfM(Structure from Motion) と呼び，これは主要な研究対象として長年扱われてきている問題である [56, 22].

膨大な研究の積み重ねによって SfM の基礎理論は固まりつつあり，その技術の一部は現実にも利用されている [2, 59]．しかし，まだまだ応用が進んでいないのは，画像特有の誤差に対するシビアさに起因しており，数ピクセルのズレで計算が破綻してしまうケースが多数存在しているためである．これは画像という二次元の情報から三次元の形状を復元する逆問題の難しさに加えて，カメラ固有の撮像系物理モデルの複雑さから来るものである．そのため，現状では決められた環境で想定されたカメラモーションにのみ対処可能なシステムが構築されていることが多く，この問題を解決することは，SfM の応用範囲を拡

大するために必要不可欠である．

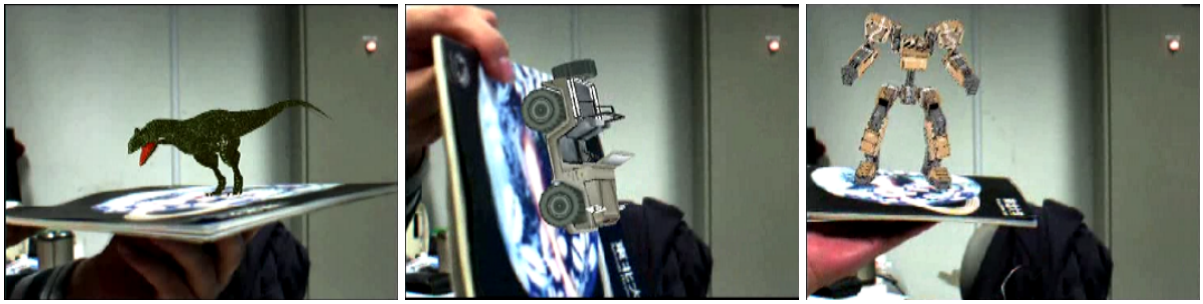


Fig. 1.1: AR（拡張現実感）の例.

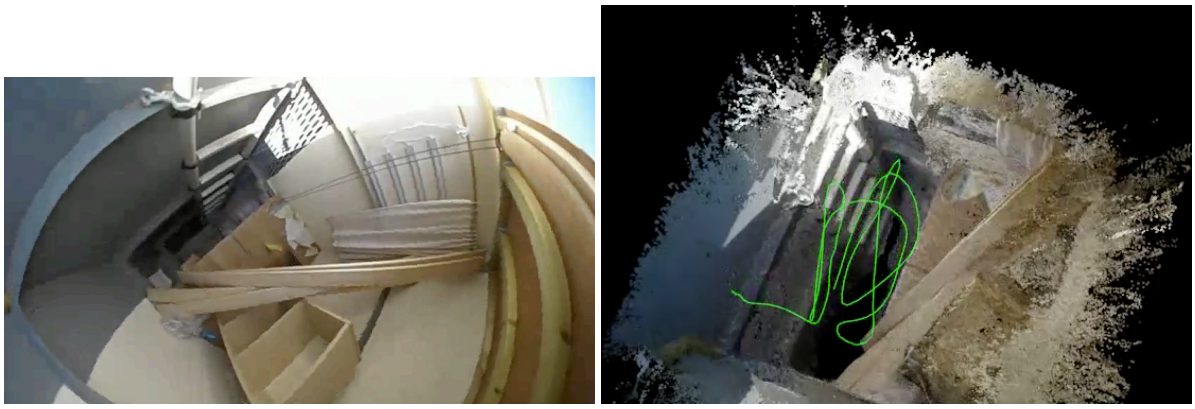


Fig. 1.2: 画像のみを使った地形測量と自己位置推定の例.

1.2　研究の目的

SfM は大別すると 2 種類に分けることができる．画像特徴を使った特徴点に基づく方法 [56, 65, 36, 1] と，画像の濃淡情報を直接使い最適化問題を解く直接法 [7, 58, 47, 18] である．これら 2 つは，長所短所が入れ替わった形になっており，それぞれに適した問題設定がある．本研究では，特徴点に基づく方法と直接法の利点を使い分け，SfM の応用範囲を拡大するために障害となる種々の問題に対処する手法の開発を目的とする．以下に SfM の際に性能劣化の原因となる問題を列挙する．

- モーションブラー
  カメラの露光時間中に各画素が読み込む位置が変わることで発生する画像のブレ（Fig.1.3）をモーションブラーと呼ぶ.

- ローリングシャッター
  CMOS センサーのカメラでは画像を行ごとに読み込むため，画像に歪み生じる現象（Fig.1.4）のことをローリングシャッターと呼ぶ.

- 画像の実効的な解像度の変化
  対象物体の距離や向きによって画像中で見える大きさが変わり，対象物体のサンプリング間隔が一定にならず実効的な解像度の変化が起こる.

- テクスチャが少ない
  画像中の濃淡の変化が少ない場合，画像特徴が多く取れないため，位置の推定精度が落ちる.

- カメラのフレームレート
  Fig.1.5 のようにフレームレートが低いカメラの場合，隣接画像間で重複する部分が減り，対応関係が付け辛くなる.

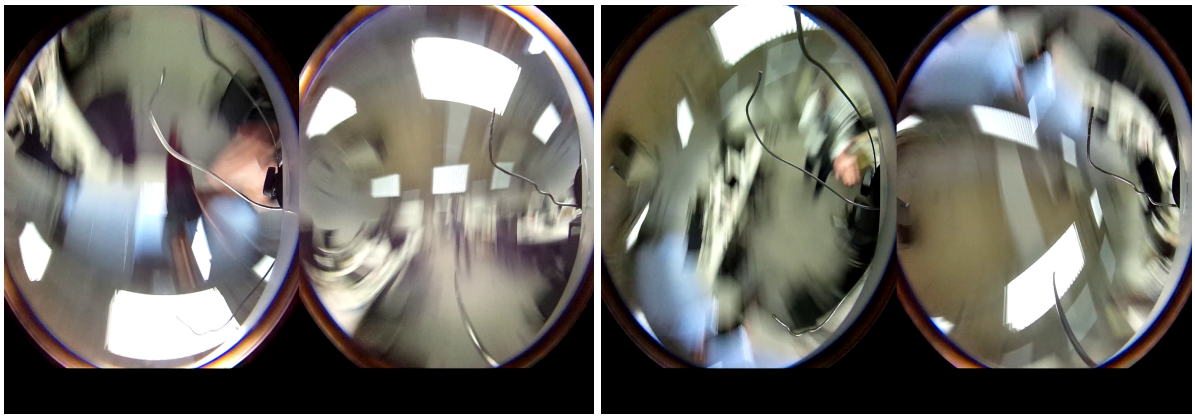


Fig. 1.3: モーションブラーによる画像のブレ.

1.3 特徴点に基づく SfM の原理

以降の節では，まず，カメラで撮像した物体がどのように画像上で現れるかを単純なピンホールカメラモデルを使って説明し，三次元空間上でカメラが動いた際の点同士の対応関係について原理を簡単に解説する．そして，画像上の対応関係から三次元復元という逆問題を解くのか，一般的な方法（8 点アルゴリズム）を要約する.

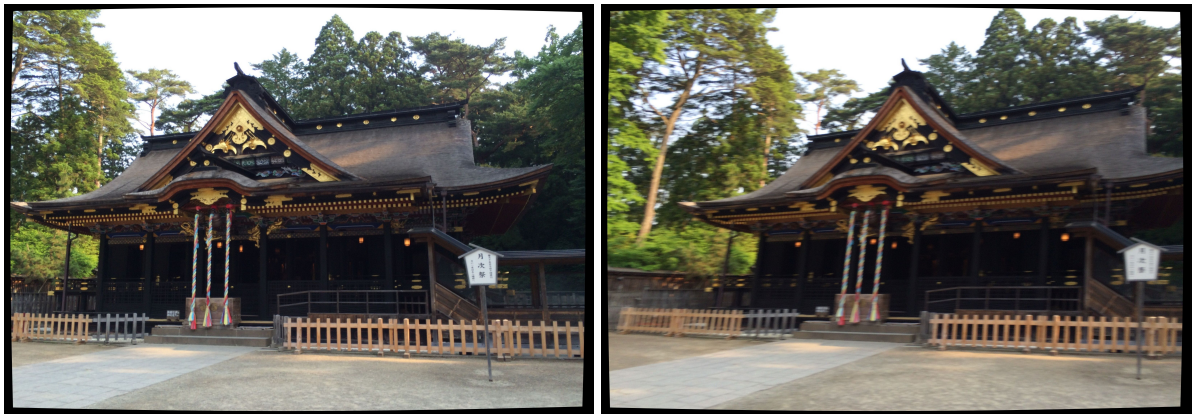


Fig. 1.4: ローリングシャッターによる画像の歪みの比較．左側が歪み無し，右側が歪みあり．

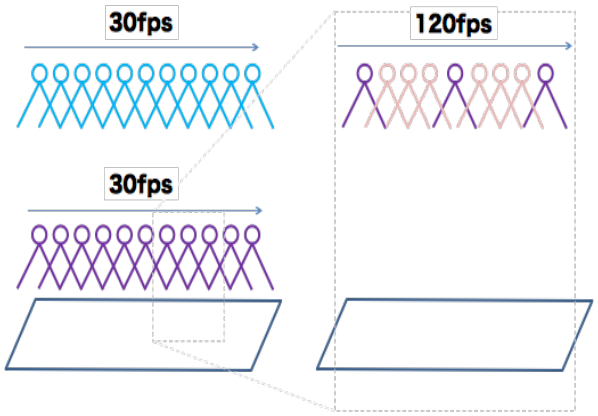


Fig. 1.5: フレームレートの違いによる画像取得間隔の比較.

1.3.1　カメラの内部パラメータ

撮像された物体が画像上でどのように表現されるのかを単純化したモデルはピンホールカメラモデルで表現される Fig.1.6．まず空間上のある 1 点を $\mathbf{X} = (x, y, z)^T$ とし，それに対応する画像上の点を $\mathbf{p} = (u, v)^T$，カメラの焦点距離を f，x と y 方向の画像中心をそれぞれ (cx, cy) と置くと，これらの対応関係は

$$\mathbf{p} = (f\frac{x}{z} + cx, f\frac{y}{z} + cy)^T. \tag{1.1}$$

これを行列の形で簡潔にまとめるため，焦点距離とカメラ中心を

$$\mathbf{K} = \begin{pmatrix} f & 0 & cx \\ 0 & f & cy \\ 0 & 0 & 1 \end{pmatrix} \tag{1.2}$$

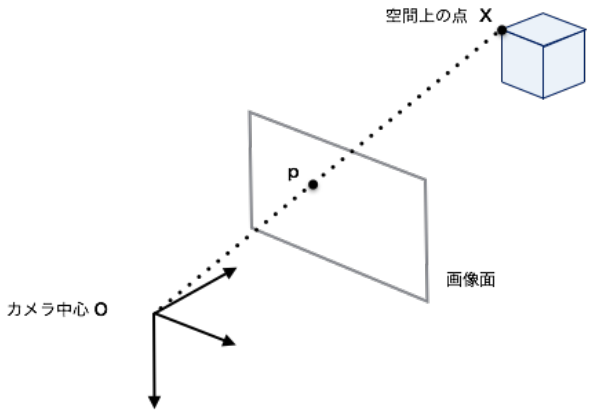


Fig. 1.6: ピンホールカメラモデル．

とする. これはカメラ毎に異なる値を持っており，内部パラメータと呼ばれている．そして $\mathbf{m} = (x/z, y/z, 1)$ と置き，さらに $\mathbf{p}$ を一次元拡張した斉次座標系を導入すると，Eq.(1.1) は

$$\mathbf{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K}\mathbf{m} \tag{1.3}$$

となる．したがって，あるカメラ座標から見たときの $\mathbf{X}$ と画像座標 $\mathbf{p}$ の関係は，内部パラメータ $\mathbf{K}$ を用いることによって表すことができる．

1.3.2　カメラの外部パラメータと 2 視点幾何

Fig.1.7 のように，カメラが空間上で動き 2 枚の画像が得られたとする．それぞれのカメラ座標系において，物体上の同じ点を $\mathbf{X}$ と $\mathbf{X}'$ とすると，2 視点間の画像の対応関係は回転行列 $\mathbf{R}$ と平行移動ベクトル $\mathbf{t}$ によって

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{t} \tag{1.4}$$

となる．$\mathbf{R} \in \mathbb{SO}(3)$ は特殊直行群であり自由度は 3，$\mathbf{t} \in \mathbb{R}^3$ で自由度は 3 である．この $\mathbf{R}$ と $\mathbf{t}$ はカメラの外部パラメータと呼ばれている．前節と同様に $\mathbf{X}$ と $\mathbf{X}'$ を斉次座標系に直しまとめると

$$\mathbf{X} = \begin{pmatrix} x & y & z & 1 \end{pmatrix}^T, \tag{1.5}$$

$$\mathbf{X}' = \begin{pmatrix} x' & y' & z' & 1 \end{pmatrix}^T, \tag{1.6}$$

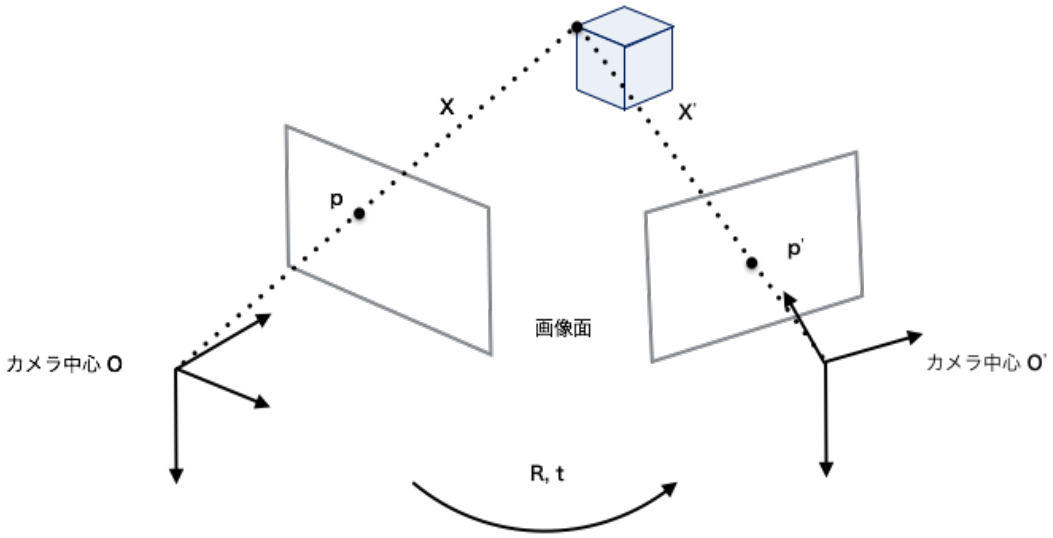


Fig. 1.7: カメラ座標の対応関係.

$$\mathbf{X}' = [\mathbf{R}|\mathbf{t}]\mathbf{X}. \tag{1.7}$$

これを前節の内部パラメータを使うと，1 番目のカメラ座標から見たときの $\mathbf{X}$ と 2 枚目の画像座標 $\mathbf{p}'$ の関係は以下の形で表すことができる．

$$\mathbf{p}' \propto \mathbf{P}\mathbf{X}. \tag{1.8}$$

$\propto$ は 0 を除く定数倍の不定性を許すという記号とする．Eq.(1.8) の $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ は (3x4) 行列であり，これは透視投影行列と呼ばれている．

1.3.3 PnP によるカメラの外部パラメータの推定

内部パラメータが与えられている時，画像内のあらかじめ空間上で座標が分かっている n 点を使ってカメラの位置・姿勢を求める問題を PnP 問題と呼ぶ．ここでは，透視投影行列の式から PnP 問題を解き，カメラの外部パラメータをどのように推定していくかを確認していく．PnP の問題では，空間上の点の座標 $\mathbf{X}$ と対応する画像座標 $\mathbf{m}$ が既知と仮定しているため，Eq.(1.8) から 1 点分の方程式は

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} \dfrac{P_{11}x + P_{12}y + P_{13}z + P_{14}}{P_{31}x + P_{32}y + P_{33}z + P_{34}} \\ \dfrac{P_{21}x + P_{22}y + P_{23}z + P_{24}}{P_{31}x + P_{32}y + P_{33}z + P_{34}} \end{pmatrix} \tag{1.9}$$

となる．1 点につき 2 つの方程式が立ち，なおかつ $\mathbf{P}$ の自由度は定数倍の不定性を除いて 11 であるため $n > 6$ を与えれば，最小二乗法などを使ってパラメータを決めることがで

きる．こうして求まった **P** を内部パラメータの逆行列を使って，$\mathbf{K}^{-1}\mathbf{P} = [\mathbf{R}|\mathbf{t}]$ と分解すれば，カメラの回転行列と平行移動ベクトルを得ることができる．

逆に，空間の点 **X** が未知で 2 枚の画像間で点の対応と **P** が分かっている場合は，空間上の座標は三角測量で求めることができる．Eq.(1.9) にそれぞれの画像点を当てはめると 4 つの方程式が立ち，求めたい自由度は x,y,z の 3 つであるため同様に計算することができる．

PnP 問題では空間上の点の座標が既知であると仮定して透視投影行列を計算し，外部パラメータを求めるが，事前知識が何もない空間ではこの仮定は成り立たない．一般的には，画像が 2 枚与えられた時に我々がわかることは，画像上で共通してあらわれる特徴的な点，例えば濃淡のエッジの差が強い点などの対応関係の計算のみである．どのようにして画像間の点の対応のみから SfM をすれば良いのかについては，次節の 8 点アルゴリズムで説明をする．

1.3.4 8 点アルゴリズム

8 点アルゴリズムの詳しい導出については省略するが，どのようにこの計算を利用して SfM に適用していくかの概要を示す．まず，画像間の点の対応関係は (3x3) の基礎行列 **F** によって以下の式で表すことができる．

$$\mathbf{p}_2^T \mathbf{F} \mathbf{p}_1 = 0. \tag{1.10}$$

ここで重要な点は，画像間の点の対応関係は 3 次元座標を介さずに **F** という式で結びつくということである．この式は，異なる内部パラメータを持った 2 台のカメラ間で成り立つ式であるが，今回は単眼の SfM に焦点を当てるため 2 枚の画像間で内部パラメータは変化しないものとする．Eq.(1.10) の両辺に定数倍しても式は成り立つことから，基礎行列の自由度は 8 である．そのため，8 つ以上の対応点から以下の式を満たすような **F** を最小二乗法などをつかって計算することができる．

$$\arg\min_F \sum_{i=0}^{n} (\mathbf{p}_{i2}^T \mathbf{F} \mathbf{p}_{i1})^2. \tag{1.11}$$

次に，基礎行列から得られるもう一つの重要な式である，エピポーラ幾何について説明する．画像 1 の任意の点 (u, v) は，画像 2 の上でどのように振る舞うかを見るために，$(a, b, c)^T = \mathbf{F}\mathbf{p}_1$ と置いて Eq.(1.10) に $\mathbf{p}_1$ を代入をすると

$$au' + bv' + c = 0 \tag{1.12}$$

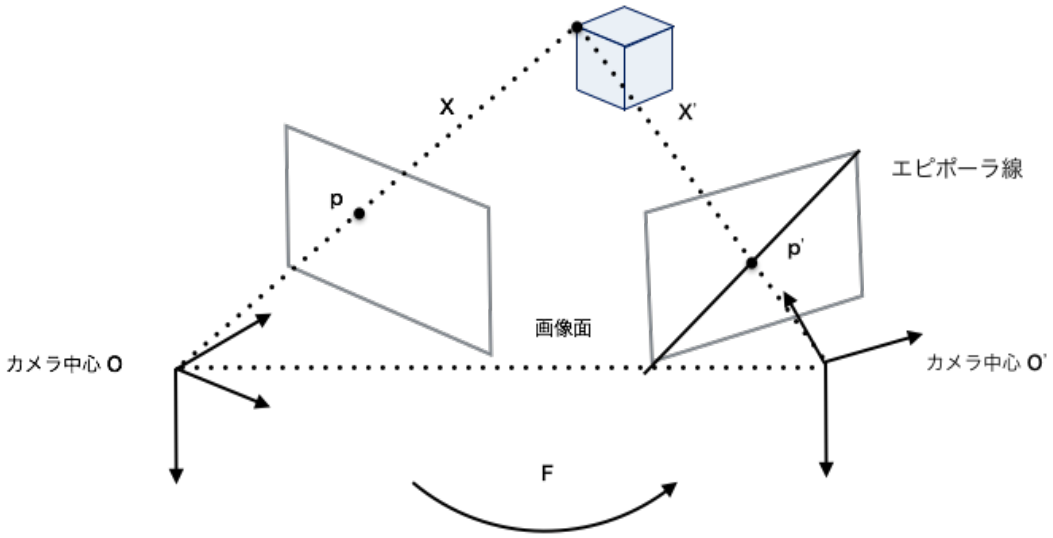


Fig. 1.8: エピポーラ幾何.

が成り立つ．(a, b, c) はそれぞれ定数で，(u', v') が 2 枚目の画像上で対応する座標になっているので，(u', v') はある直線上に乗ることがわかる（Fig.1.8）．この直線のことをエピポーラ線と呼び，点の対応がエピポーラ拘束によって支配されていることがわかる．特徴点ベースの方法では，対応する点の外れ値処理をする際にエピポーラ線と特徴点の距離を評価として用いることが多い．

基礎行列は，Eq.(1.13) のように内部パラメータと基本行列と呼ばれるものの組み合わせでできており，基本行列は 2 枚のカメラ間の回転行列と平行移動ベクトルの歪み対称行列の積の形なっている．

$$\mathbf{K}^{-T}\mathbf{F}\mathbf{K}^{-1} = \mathbf{E} = [\mathbf{t}]_X\mathbf{R}. \tag{1.13}$$

したがって，基礎行列を内部パラメータを使って基本行列に直し，特異値分解をすることで $\mathbf{R}$ と $\mathbf{t}$ を導出することができる．この分解によって符号と回転の反転を含む 4 つの解の候補である $\mathbf{R}$ と $\mathbf{t}$ が得られるが，そのうちの 3 つは偽の解である．前節の三角測量の原理から，全ての点がカメラの正面にあるという条件を使うことで 1 つに解を定めることが可能である．以上が基本的な SfM の手順になっており，2 枚の画像の点の対応からカメラ位置・姿勢と三次元点の座標が導出される．

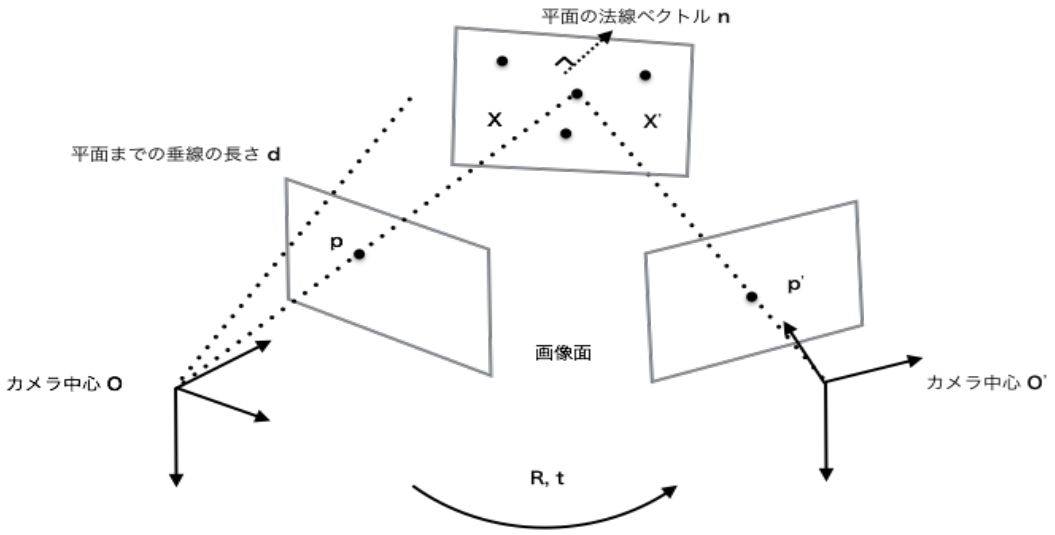


Fig. 1.9: すべての点が平面に乗っている場合.

1.3.5 二次元射影変換

この節では，SfM の特殊なケースであるすべての点が同じ平面上に乗っている場合について解説していく．まず，Fig.1.9 の片方のカメラ O から見たときの平面の法線ベクトルを **n**（長さ 1 の単位法線ベクトル），O から平面へ下ろした垂線の長さを d とすると

$$\mathbf{n}^T X = d. \tag{1.14}$$

が成り立つ．$\frac{\mathbf{n}^T X}{d} = 1$ の条件を使い Eq.(1.4) に代入すると

$$\mathbf{X}' = (\mathbf{R} + \frac{\mathbf{t}\mathbf{n}^T}{d})\mathbf{X} \tag{1.15}$$

となる．$\mathbf{X} = z\mathbf{m}$，$\mathbf{X}' = z'\mathbf{m}'$ であることと，Eq.(1.3) を使うこと以下の式が導かれる．

$$\frac{z'}{z}\mathbf{p}' = \mathbf{K}(\mathbf{R} + \frac{\mathbf{t}\mathbf{n}^T}{d})\mathbf{K}^{-1}\mathbf{p} \tag{1.16}$$

$$\mathbf{p}' \propto \mathbf{H}\mathbf{p} \tag{1.17}$$

$$\mathbf{H} = \mathbf{K}(\mathbf{R} + \frac{\mathbf{t}\mathbf{n}^T}{d})\mathbf{K}^{-1} \tag{1.18}$$

以上のようにカメラの内部パラメータと回転・平行移動の外部パラメータがわかってて，なおかつ対象平面の法線ベクトルと平面までの距離が既知であれば，画像上の点の対応関係は一意に定まると言える．このような **H** のことを二次元射影変換行列，または単にホモグラフィー行列と呼ぶ．二次元射影変換行列は (3x3) で，定数倍の不定性があるため自由度は 8 である．

画像間の点の対応からHを直接求める方法については，PnPの問題の時と同様に解くことができる．1つの点の対応から

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} \dfrac{H_{11}u + H_{12}v + H_{13}}{H_{31}u + H_{32}v + H_{33}} \\ \dfrac{H_{21}u + H_{22}v + H_{23}}{H_{31}u + H_{32}v + H_{33}} \end{pmatrix} \tag{1.19}$$

2本の方程式ができ，自由度は8であるため4つの対応点が分かっていればHを計算することができる．単一平面の対応からSfMをする手順は，法線**n**と内部パラメータが既知の状態で二次元射影変換を計算し，それを分解することによってカメラの回転行列と平行移動ベクトル（カメラOから平面までの距離dは平行移動ベクトルの中に含まれた形になる）が得られる．

1.3.6 多視点幾何によるバンドル調整

前節までは2視点におけるSfMを取り扱ったが，ここではカメラ数が3，4…，nと増えていった場合の計算手法について述べる．まず最も単純なケースとしてカメラ数の増加に合わせて2視点SfMのつなぎ合わせをすることを考える．初期の2枚の画像では，**R**, **t**, **X**がそれぞれ未知数であるため，特徴点を使って画像間の点の対応を求め，8点アルゴリズムなどで幾何学パラメータを推定する．3枚目以降の画像では，既知となった**X**を使ってPnPをすることで**R**, **t**を計算し，未知の**X**については三角測量で求める．この際に問題となるのが，画像の枚数が増えるにしたがって復元される点の誤差が蓄積し，それに引きづられる形で**R**, **t**にズレが生じるということである．これは特徴点の観測誤差や透視投影行列の推定誤差，用いる内部パラメータ自体の誤差が複雑に絡み合うことで起こる問題である．しかしながら，複数の画像にまたがって同一の点が観測されている場合の全体最適化手法については，コンピュータビジョンの分野では古くから研究されており，バンドル調整と呼ばれている．この節では，バンドル調整で複数枚の点の対応をまとめて最適化する方法について説明をしていく．

空間上を動くカメラによってn枚の画像が撮影され，それぞれの画像上に共通する点（特徴点など）がm個観測されている状況を想定する．ここで，各カメラの透視投影行列$\mathbf{P}_i(i = 0, ..., n)$とすると

$$\mathbf{P}_i = \mathbf{K}[\mathbf{R}_i|\mathbf{t}_i] \tag{1.20}$$

$$\mathbf{P}_i = \mathbf{P}(\mathbf{K}, \mathbf{R}_i, \mathbf{t}_i) \tag{1.21}$$

と書ける．ここで観測点の番号を $j = 0, ..., m$ と置き，各点の画像上の座標を $\mathbf{p}_{ij} = (u_{ij}, v_{ij})^T$，空間上の座標を $\mathbf{X}_j = (x_j, y_j, z_j)^T$ とすると，画像座標と空間上の座標の関係は以下の通りになる．

$$\begin{pmatrix} \mathbf{p}_{ij} \\ 1 \end{pmatrix} \propto \mathbf{P}(\mathbf{K}, \mathbf{R}_i, \mathbf{t}_i) \begin{pmatrix} \mathbf{X}_j \\ 1 \end{pmatrix} \tag{1.22}$$

この関係を使って空間上の点をすべての画像に再投影し，観測点との誤差を最小化すべき評価関数とすると

$$E(\mathbf{R}_0, ..., \mathbf{R}_n, \mathbf{t}_0, ..., \mathbf{t}_n, \mathbf{X}_0, ..., \mathbf{X}_m) = \frac{1}{2} \sum_{i,j} (u_{ij} - u(\mathbf{K}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j))^2 + (v_{ij} - v(\mathbf{K}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j))^2 \tag{1.23}$$

となる．これは非線形方程式のため，十分に真値に近い $\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j$ を初期値に，ガウスニュートン法やレベンバーク・マルカート法などの反復解法を用いて解くことができる．初期値としては，2 視点 SfM の出力を与えることが一般的である．バンドル調整では以上のように全ての観測を使って最適化することで，SfM の精度を向上させることができる．だが，あくまで Eq.(1.23) は全ての観測が理想的な環境で得られることを想定しているため，ローリングシャッターやモーションブラーといった外乱が発生した場合には頑健とは言い難い．

1.4 直接法による平面追跡の原理

画像上で平面物体を追跡することは，コンピュータビジョンにおいて重要な処理の一つであり，様々な応用に用いられている．よく知られている応用例として，ある平面領域を追跡し，そこからカメラと平面との相対的な位置関係はどのようになっているか，つまり，平面に対するカメラの回転パラメータと並進パラメータを推定することで，視覚的にロボットの制御をするビジュアルサーボや，画像上に CG などの仮想物体を重畳する AR (拡張現実感) が挙げられる．また，Lucas-Kanade 法の特徴点追跡に代表されるように，物体表面を微小な平面の集合として近似し，それを三次元空間上で追跡することで，カメラ運動や対象の幾何学形状を復元する SFM でも用いられる．

上記のアプリケーションにおいて特に必要とされるのは，精度・高速性・安定性であり，当然ながら平面追跡の性能の良し悪しが，アプリケーションの完成度に大きな影響を及ぼす．そのため，コンピュータビジョンの黎明期から，数多くの研究者達がこの根幹技術である平面追跡のため手法の開発や改良を行い，成果をあげてきた．しかし，まだいくつかの問題が残されており，盛んに研究が進められている．

1.4.1 平面追跡の関連研究

平面追跡のための方法は，これまで多数提案されてきた．それらは大まかに 2 つのカテゴリーに分類できる．一つは，画像から点や線分，閉曲線といった原始的な特徴を取り出し，これらに基づいて平面を追跡するものである（代表的なものに [37, 48, 28, 52]）．

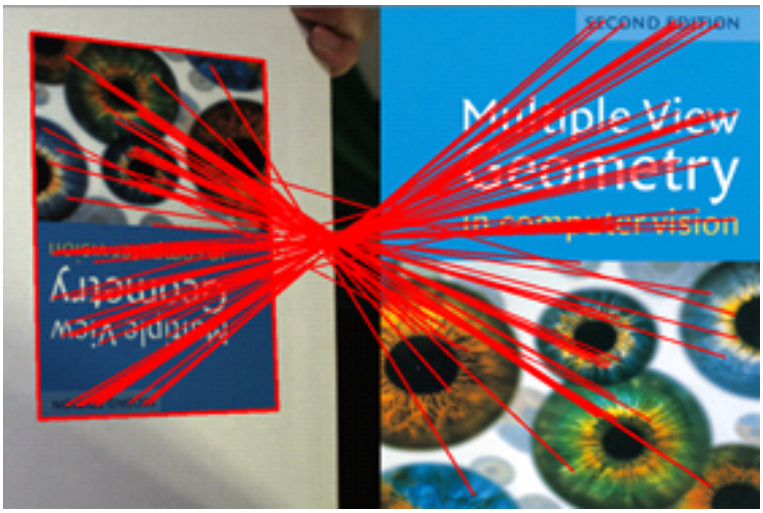


Fig. 1.10: SIFT 特徴量を用いた画像間の点の対応付け．

もう一つは，そのような特徴抽出を行わず，平面の見えを直接用いる方法である（[7, 41, 8, 9, 44, 15, 16, 13]）．こちらは，Lucas-Kanade 法に見られるように，濃淡の差の二乗和である SSD(Sum of Squared Differences) などを，見えの近さの尺度，すなわち画像同士の類似度とし，変形のパラメータを未知の変数と置き，画像類似度が最大となる位置を計算している．一般的には，Fig.1.4.1 のように，ニュートン法（あるいはその変種）を用いて，画像がどのように微小変形したかを推定し，その微小変形の積み重ねによって，平面の画像上の姿勢を定めるている．

本論文では後者の方法に重点をおいて考える．多くの研究がある（[7, 41, 8, 9, 44, 15, 16, 13]）が，それぞれ異なる問題意識や動機に基づいている．まず，最適化の方法に関する研究がいくつかある．Baker らは，最適化の計算方法を分類し，現在の推定値まわりでの撮影画像の変動に基づく反復計算を行う forward 法に比べて，テンプレートの変動に基づく inverse 法の方が，より効率的であると述べている [7]．Malis らは，この反復計算で用いるヘッセ行列を，一般的なガウスニュートン近似よりも高精度かつ小さな計算量で計算し得る ESM(Efficient Second-order Method) 法を提案している [41, 9]．また，最適化の目的関数となるテンプレートと撮影画像の近さの尺度を，改善しようとする研究もある．

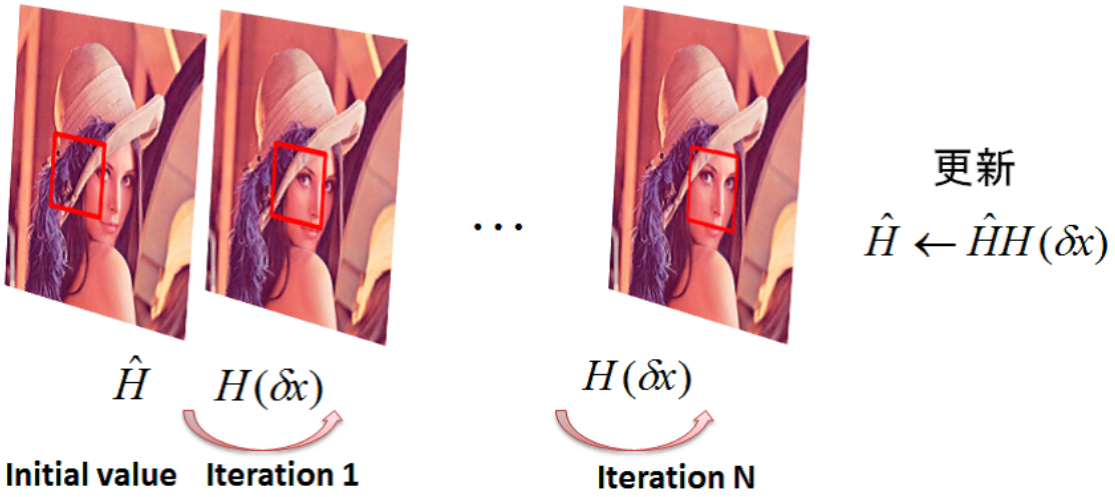


Fig. 1.11: ニュートン法による平面領域の位置推定.

SSD を利用する従来の方法に対し，最近，相互情報量 (Mutual Information) に基づく尺度の導入が提案されている [15, 16, 13]．相互情報量に基づく方法では，テンプレートと撮影画像の様相がある程度変化しても，類似度が高くなるため，Fig.1.12 のように，色調が反転した画像に対しても追跡が可能である．

一方，最適化の方法の研究とは別に，平面追跡を難しくする要因への対処方法も研究されている．まず，平面への照明の当たり方が変化すると，その見えは幾何学的に変化するだけでなく濃淡値そのものが変化し，これによって追跡性能が低下する．Silveira らは，ESM 法の枠組みで照明変動を扱える方法を示している [57]．Dame らも，相互情報量を用いる方法の研究 [13] において照明変動への対処を議論している．また，カメラのシャッター速度に比して平面が高速に動く場合には，モーションブラーが発生し，これも追跡性能を低下させる．これへの対策が Jin ら [31]，そして Mei ら [44] によって示されている．Park らは，モーションブラーが直線的な平面の運動によって生成される場合，わずかに ESM 法を修正するだけでモーションブラーが扱える方法を提示している [49].

1.4.2　特徴ベースの平面追跡手法

画像から点や線分，閉曲線といった原始的な特徴を取り出し，これらに基づいて追跡する特徴ベースの手法の利点と欠点についてまとめる．

特徴ベースの方法は，フレーム毎に画像から特徴を抜き出し，点やエッジなどの疎な対応から，射影変換のパラメータを推定している．そのため，一部の対応点がオクルージョ

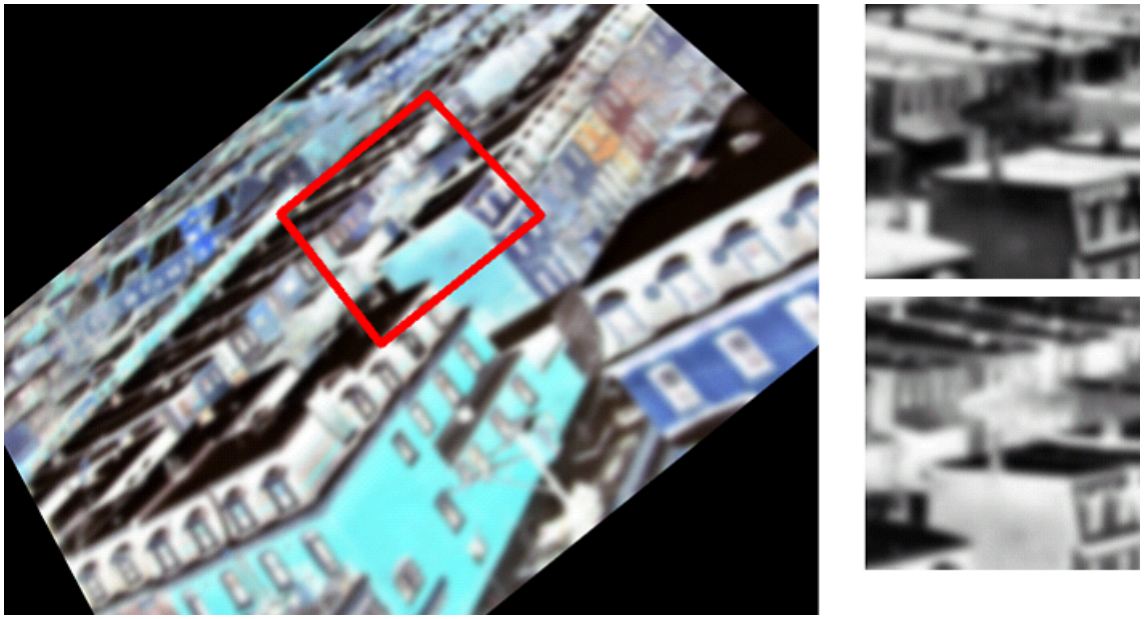


Fig. 1.12: 相互情報量を類似度にした場合のトラッキングの様子．右下が推定された領域．テンプレートと追跡対象の色が反転しても追跡が可能である．

ンなどによって隠れてしまった場合でも，ほかの対応点によって推定を行うことができる．さらに，フレーム間の連続性を仮定する必要が無く，対象がフレーム間で大きく動いた場合でも追跡することができる．

短所として，推定精度を向上させるために多数の対応点が必要であったり，特徴点抽出や対応付けに時間がかかることが挙げられる．これらは，平面が大きく傾いたり，遠くに離れていく場合に特に問題視される．Fig.1.4.2 はアフィン変換の影響を考慮した SIFT 特徴量を用い，マッチングを行った結果である [45]．この図では，平面領域が小さく見えているため，十分な数の対応点が得られていないことがわかる．

1.4.3　直接法による平面追跡手法

特徴抽出を行わず，平面の見えを直接用いて追跡を行う，直接法による追跡手法についてまとめる．

この追跡手法は、すべてのピクセルを密に幾何学量の推定に用いているため高精度である．また，平面が大きく傾いた状態でも，特徴ベースの方法に比べて追跡性能が良い傾向がある．計算速度の面では，Baker らの提案した Inverse Compositional 法が Forward 法に比べてはるかに良く，[43] の報告によると，評価関数が SSD の場合，約 2 倍程度速くなっ

Fig. 1.13: 平面が大きく傾いたときの ASIFT の振る舞い．平面領域が小さく見えるため，対応付られる特徴点の数が少なくなり，精度が悪化する．

ている．しかし，この方法は，ニュートン法の計算においてヘッセ行列を定数にしているため，照明変化やモーションブラーへの拡張をすることができない．

1.4.4　問題の定式化

現在の画像を $\mathcal{I}$，追跡対象となるテンプレートを $\mathcal{I}^*$ と書く．テンプレート $\mathcal{I}^*$ は，$\mathcal{I}$ より小サイズの画像で，追跡対象とする平面の矩形領域を真正面からカメラで撮影したものに相当する．カメラと平面の相対位置姿勢が変化し，それにともなって $\mathcal{I}$ が変化するとき，その上で平面を追跡したい．

テンプレート $\mathcal{I}^*$ とそれを撮影した画像 $\mathcal{I}$ に対し，Fig.1.4.4 のように $\mathcal{I}^*$ の点 $\mathbf{p}^*$ を $\mathcal{I}$ の点 $\mathbf{p}$ に写す変換（warp）を

$$\mathbf{p} = w(\mathbf{p}^*) \tag{1.24}$$

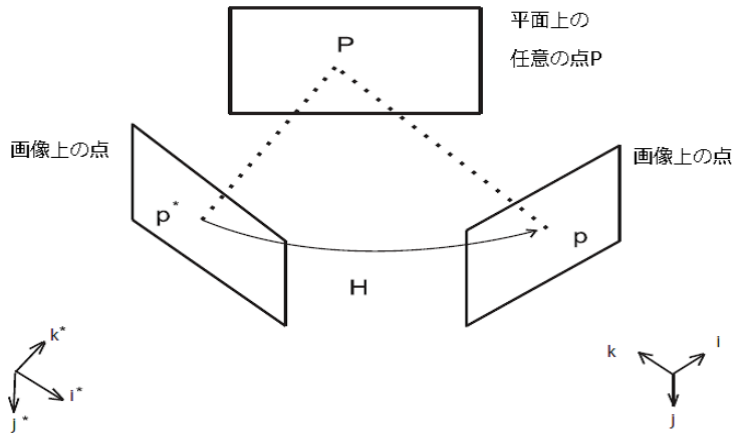


Fig. 1.14: 三次元上にある平面パターンの任意の点を異なる視点から撮影したときに与えられる二次元射影変換.

と表す．この変換は理想的な透視カメラを仮定すると，次の 2 次元射影変換で与えられる．

$$\mathbf{p} = w(\mathbf{p}^*; \mathbf{H}) \propto \mathbf{H}\mathbf{p}^* \tag{1.25}$$

$\mathbf{H}$ は 2 次元射影変換を表す 3×3 行列で，$\mathbf{p}$ および $\mathbf{p}^*$ はそれぞれ $\mathcal{I}$ および $\mathcal{I}^*$ 上の画像座標の同次表現（3 次元ベクトル）であり，$\propto$ は両辺のベクトルが長さを除いて等しいことを表す．なおこれ以降 $\mathbf{p}$ と $\mathbf{p}^*$ それぞれ，文脈に応じて同じ画像の点についての同次座標と非同次座標のどちらかを表すものとする．

今，追跡対象の平面の同一点の画像上の明るさが，平面姿勢によらず変化しないとすれば，平面を撮影した画像 $\mathcal{I}(\mathbf{p})$ とテンプレート $\mathcal{I}^*(\mathbf{p}^*)$ は次のような関係を持つことになる：

$$\mathcal{I}(w(\mathbf{p}^*; \mathbf{H})) = \mathcal{I}^*(\mathbf{p}^*). \tag{1.26}$$

このことから，式の両辺の差が最小となるように $\mathbf{H}$ を定めることと考える．この最小化はニュートン法系の反復計算によって行う．具体的には，現在の推定値 $\hat{\mathbf{H}}$ が与えられたとき，

$$\mathbf{H}_o = \underset{\mathbf{H}}{\operatorname{argmin}} \sum_i \left[\mathcal{I}(w(\mathbf{p}_i^*; \hat{\mathbf{H}}\mathbf{H})) - \mathcal{I}^*(\mathbf{p}_i^*) \right]^2 \tag{1.27}$$

なる増分 $\mathbf{H}_o$ を求める（なおこの式で $\mathbf{H}$ を再定義しており，$\hat{\mathbf{H}}\mathbf{H}$ が Eq.(1.26) の $\mathbf{H}$ に相当することに注意する）．その後，$\hat{\mathbf{H}} \leftarrow \hat{\mathbf{H}}\mathbf{H}_o$ と更新し，以降これを繰り返す.（Baker らの分類 [7] によれば，これは forward compositional 法にあたる.）新しい画像フレームを処理するときは，前フレームでの推定値 $\hat{\mathbf{H}}$ をそのまま使って（つまり初期値として），この反復計算を収束するまで繰り返す.

ここで増分 $\mathbf{H}$ は 3×3 行列で 9 成分あるが，スケール倍の不定性があるため，その自由度は 8 しかなく，パラメータの取り方が重要である．よい方法は，$\mathbf{H}$ が特殊線形群 $\mathbb{SL}(3)$ に属するようにパラメータをとることで，具体的には 8 次元ベクトル $\mathbf{x}$ によって $\mathbf{H}=\mathbf{H}(\mathbf{x})$ を指数写像を用いて

$$\mathbf{H}(\mathbf{x})=\exp\left(\sum_{j=1}^{8}x_j\mathbf{G}_j\right) \tag{1.28}$$

と表現することである（[8, 9]）．ここで $\mathbf{G}_j$ $(j=1,\ldots,8)$ はリー代数の生成作用素となる 3×3 行列である．$\mathbf{H}$ をこのように表現することで $\det\mathbf{H}=1$ が約束され，$\mathbf{H}$ のスケール倍の不定性を除去している．この表現に基づき，反復の毎ステップで

$$\mathbf{x}_o=\underset{\mathbf{x}}{\operatorname{argmin}}\sum_i\left[\mathcal{I}(\hat{\mathbf{H}}\mathbf{H}(\mathbf{x})\mathbf{p}_i^*)-\mathcal{I}^*(\mathbf{p}_i^*)\right]^2 \tag{1.29}$$

を求め，$\hat{\mathbf{H}}\leftarrow\hat{\mathbf{H}}\mathbf{H}(\mathbf{x}_o)$ と更新し，これを繰り返す．

1.4.5 平面追跡手法の分類

Lucas-Kanade 法に代表される，アピアランスベースの平面追跡手法は，大きく分けて Forward Additive(FA) 法，Forward Compositional(FC) 法，Inverse Additive(IA) 法，Inverse Compositional(IC) 法の 4 つに分類される．これらの方法は，コスト関数の構成方法とパラメータの更新方法が異なっており，計算量に劇的な違いがみられる．以降より，代表的な FC 法と IC 法について触れ，それを一般化した ESM 法を紹介する．

Forward Compositional Algorithm

$$\mathbf{x}_o=\underset{\mathbf{x}}{\operatorname{argmin}}\sum_i\left[\mathcal{I}(\hat{\mathbf{H}}\mathbf{H}(\mathbf{x})\mathbf{p}_i^*)-\mathcal{I}^*(\mathbf{p}_i^*)\right]^2 \tag{1.30}$$

FC 法は，上の式のコスト関数を計算し，推定値を反復ごとに更新してテンプレート画像と一致する領域を見つけている．この関数は一般的に非線形であるため，初期値周り（つまり $\mathbf{H}(\mathbf{0})=\mathbf{I}$）でテイラー展開すると

$$\sum_i\left[\mathcal{I}(w(w(\mathbf{p}_i^*;\mathbf{I});\hat{\mathbf{H}}))+\nabla\mathcal{I}(w(\mathbf{p}_i^*;\hat{\mathbf{H}}))\frac{\partial w}{\partial\mathbf{H}}\frac{\partial\mathbf{H}(\mathbf{x})}{\partial\mathbf{x}}\mathbf{x}-\mathcal{I}^*(\mathbf{p}_i^*)\right]^2 \tag{1.31}$$

となる．$\nabla\mathcal{I}(w(\mathbf{p}_i^*;\hat{\mathbf{H}}))$ は，初期値 $\hat{\mathbf{H}}$ をつかってテンプレートの形に戻した画像 $\mathcal{I}(w(\mathbf{p}_i^*;\hat{\mathbf{H}}))$ を，画像座標 u，v で空間微分したベクトルである．

$$\nabla \mathcal{I}(w(\mathbf{p}_i^*; \hat{\mathbf{H}})) = \left[\frac{\partial I(w(\mathbf{p}_i^*; \hat{\mathbf{H}}))}{\partial u}, \ \frac{\partial I(w(\mathbf{p}_i^*; \hat{\mathbf{H}}))}{\partial v} \right] \tag{1.32}$$

次にウォープの関数 $w = (w_u, w_v)^\top$ を二次元射影変換のパラメータで微分した行列は

$$\frac{\partial w}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial w_u}{\partial x_1} & \frac{\partial w_u}{\partial x_2} & \cdots & \frac{\partial w_u}{\partial x_8} \\ \frac{\partial w_v}{\partial x_1} & \frac{\partial w_v}{\partial x_2} & \cdots & \frac{\partial w_v}{\partial x_8} \end{bmatrix} \tag{1.33}$$

となる．二次元射影変換 $\mathbf{H}(\mathbf{x})$ をパラメータ x について微分すると Eq.(1.28) の特殊線形群の基底を並べた定数行列が計算される．Eq.(1.31) が最小となる解 $\mathbf{x}_o$ は，Eq.(1.31) を微分して停留点となる位置を計算することで得られる．

$$\mathbf{x}_o = \mathbf{S}_{FC}^{-1} \sum_i \left[\nabla \mathcal{I}(w(\mathbf{p}_i^*; \hat{\mathbf{H}})) \frac{\partial w}{\partial \mathbf{H}} \frac{\partial \mathbf{H}(\mathbf{x})}{\partial \mathbf{x}} \right]^T \left[\mathcal{I}^*(\mathbf{p}_i^*) - \mathcal{I}(w(\mathbf{p}_i^*; \hat{\mathbf{H}})) \right] \tag{1.34}$$

ここで $\mathbf{S}_{FC}$ はガウスニュートン法における 8×8 のヘッセ行列である．

$$\mathbf{S}_{FC} = \sum_i \left[\nabla \mathcal{I}(w(\mathbf{p}_i^*; \hat{\mathbf{H}})) \frac{\partial w}{\partial \mathbf{H}} \frac{\partial \mathbf{H}(\mathbf{x})}{\partial \mathbf{x}} \right]^T \left[\nabla \mathcal{I}(w(\mathbf{p}_i^*; \hat{\mathbf{H}})) \frac{\partial w}{\partial \mathbf{H}} \frac{\partial \mathbf{H}(\mathbf{x})}{\partial \mathbf{x}} \right] \tag{1.35}$$

Inverse Compositional Algorithm

IC 法は，Eq.(1.27) とは立式がことなっており，Eq.(1.36) のようにテンプレート側に推定すべきパラメータを追加している．

$$\mathbf{H}_o = \underset{\mathbf{H}}{\operatorname{argmin}} \sum_i \left[\mathcal{I}^*(w(\mathbf{p}_i^*; \mathbf{H}(\mathbf{x}))) - \mathcal{I}(w(\mathbf{p}_i^*; \hat{\mathbf{H}})) \right]^2 \tag{1.36}$$

これを FC 法と同様に，初期値周りでテイラー展開すると

$$\sum_i \left[\mathcal{I}^*(w(\mathbf{p}_i^*; \mathbf{I})) + \nabla \mathcal{I}^* \frac{\partial w}{\partial \mathbf{H}} \frac{\partial \mathbf{H}(\mathbf{x})}{\partial \mathbf{x}} \mathbf{x} - \mathcal{I}(w(\mathbf{p}_i^*; \hat{\mathbf{H}})) \right]^2 \tag{1.37}$$

となり，この関数を最小にする $\mathbf{x}_o$ を求めると以下のようになる．

$$\mathbf{x}_o = \mathbf{S}_{IC}^{-1} \sum_i \left[\nabla \mathcal{I}^* \frac{\partial w}{\partial \mathbf{H}} \frac{\partial \mathbf{H}(\mathbf{x})}{\partial \mathbf{x}} \right]^T \left[\mathcal{I}(w(\mathbf{p}_i^*; \hat{\mathbf{H}})) - \mathcal{I}^*(\mathbf{p}_i^*) \right] \tag{1.38}$$

$$\mathbf{S}_{IC} = \sum_i \left[\nabla \mathcal{I}^* \frac{\partial w}{\partial \mathbf{H}} \frac{\partial \mathbf{H}(\mathbf{x})}{\partial \mathbf{x}} \right]^T \left[\nabla \mathcal{I}^* \frac{\partial w}{\partial \mathbf{H}} \frac{\partial \mathbf{H}(\mathbf{x})}{\partial \mathbf{x}} \right] \tag{1.39}$$

ヘッセ行列がテンプレート画像の微分に基づいており，事前に計算できる点が FC 法と大きく異なっている．さらに，IC 法はテンプレートからの微小変形量を推定しているため，ホモグラフィの更新方法は Eq.(1.40) となる．

$$\hat{\mathbf{H}} \leftarrow \hat{\mathbf{H}} \mathbf{H}(\mathbf{x}_o)^{-1} \tag{1.40}$$

Efficient Second-order Minimization

$\mathbf{y}(\mathbf{x})$ を，FC 法における画素ごとの輝度値の差分を格納した $q \times 1$ のベクトルとすると

$$\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}) \quad y_2(\mathbf{x}) \quad \dots \quad y_q(\mathbf{x})]^\top \tag{1.41}$$

と置くことができる．

$\mathbf{x} = \mathbf{0}$ の近傍で二階のテイラー展開をすると

$$\mathbf{y}(\mathbf{x}) = \mathbf{y}(\mathbf{0}) + \mathbf{J}(\mathbf{0})\mathbf{x} + \frac{1}{2}\mathbf{M}(\mathbf{0}, \mathbf{x})\mathbf{x} \tag{1.42}$$

となる．ここで $\mathbf{J}(\mathbf{x})$ は $q \times 8$ のヤコビ行列 Eq.(1.43)，$\mathbf{M}(\mathbf{x}_1, \mathbf{x}_2)$ は以下のような $q \times 8$ の行列で定義される．

$$\mathbf{J}(\mathbf{x}) = \nabla_{\mathbf{x}} \mathbf{y}(\mathbf{x}) \tag{1.43}$$

$$\mathbf{M}(\mathbf{x}_1, \mathbf{x}_2) = \nabla_{\mathbf{x}_1} (\mathbf{J}(\mathbf{x}_1)\mathbf{x}_2) \tag{1.44}$$

そして Eq.(1.42) の SSD コスト関数を

$$f(\mathbf{x}) = \frac{1}{2} \| \mathbf{y}(\mathbf{0}) + \mathbf{J}(\mathbf{0})\mathbf{x} + \frac{1}{2}\mathbf{M}(\mathbf{0}, \mathbf{x})\mathbf{x} \|^2 \tag{1.45}$$

とすると，この関数を最小にする $\mathbf{x} = \mathbf{x}_0$ は，ニュートン法では

$$\mathbf{x}_0 = -\mathbf{S}^{-1} \mathbf{J}(\mathbf{0})^T \mathbf{y}(\mathbf{0}) \tag{1.46}$$

$$\mathbf{S} = \mathbf{J}(\mathbf{0})^T \mathbf{J}(\mathbf{0}) + \sum_{i=0}^{q} \left. \frac{\partial^2 y_i(\mathbf{x})}{\partial x^2} \right|_{\mathbf{x}=\mathbf{0}} y_i(\mathbf{0}) \tag{1.47}$$

と求まるが，ヘッセ行列 $\frac{\partial^2 y_i(\mathbf{x})}{\partial x^2}$ の計算コストが高いため，一般的には $\mathbf{S}$ を以下のように置き換えることで計算量を減らしている．

・ 降下法

$$\mathbf{S} = \alpha \mathbf{I} \qquad \alpha > 0 \tag{1.48}$$

・ ガウス-ニュートン法

$$\mathbf{S} = \mathbf{J}(\mathbf{0})^T \mathbf{J}(\mathbf{0}) \tag{1.49}$$

・ レベンバーグ・マルカート法

$$\mathbf{S} = \mathbf{J}(\mathbf{0})^T\mathbf{J}(\mathbf{0}) + \alpha\mathbf{I} \qquad \alpha > 0 \tag{1.50}$$

ESM では，ヘッセ行列を計算することなく，効率的に二階の項まで近似している．まず，ヘッセ行列が出てくる原因であった $\mathbf{M}(\mathbf{0},\mathbf{x})$ を消去するために，$\mathbf{J}(\mathbf{x})$ を Eq.(1.51) のように二階の項までテイラー展開し，Eq.(1.42) へ代入すると

$$\mathbf{J}(\mathbf{x}) = \mathbf{J}(\mathbf{0}) + \mathbf{M}(\mathbf{0},\mathbf{x}) \tag{1.51}$$

$$\mathbf{y}(\mathbf{x}) = \mathbf{y}(\mathbf{0}) + \frac{1}{2}(\mathbf{J}(\mathbf{0}) + \mathbf{J}(\mathbf{x}))\mathbf{x} \tag{1.52}$$

Eq.(1.52) が成り立つ．$\mathbf{H}(\mathbf{x})$ は Eq.(1.28) のように，リー代数の基底を使って計算することで，ホモグラフィを特殊線形群 $\mathbb{SL}(3)$ に属するような行列に拘束することができ，その基底は以下の形で記述される．

$$\mathbf{A}_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{A}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{A}_3 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{A}_4 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{A}_5 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{A}_6 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{A}_7 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \qquad \mathbf{A}_8 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

指数写像の性質により，Eq.(1.52) の $\mathbf{J}(\mathbf{0}) + \mathbf{J}(\mathbf{x})$ に，真の解 $\mathbf{x} = \mathbf{x_o}$ を代入したものは，以下のように変形することができる．

$$\mathbf{J}(\mathbf{0}) + \mathbf{J}(\mathbf{x}_o) = \mathbf{J}_I\mathbf{J}_w\mathbf{J}_\mathbf{H} + \mathbf{J}_{I^*}\mathbf{J}_w\mathbf{J}_\mathbf{H} \tag{1.53}$$

J_I と J_{I^*} はそれぞれホモグラフィで変換した画像の空間微分，テンプレート画像空間微分である．$\mathbf{J}_w$ はウォープの関数の微分であり，テンプレート画像の位置情報が格納されて

いる．$\mathbf{J_H}$ は特殊線形群を構成するリー代数の基底が格納された行列である．そして $\mathbf{J}_{esm}$ を以下のようにまとめると

$$\mathbf{J}_{esm} = \frac{1}{2}(\mathbf{J}_I + \mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_H \tag{1.54}$$

最小化すべきコスト関数 $f(\mathbf{x})$ は，Eq.(1.55) で与えられる．

$$f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y}(\mathbf{0}) + \mathbf{J}_{esm}\mathbf{x}\|^2 \tag{1.55}$$

これを最小にする $\mathbf{x} = \mathbf{x}_o$ は

$$\mathbf{x}_o = -\mathbf{J}^+_{esm}\mathbf{y}(\mathbf{0}) \tag{1.56}$$

と求まる．$\mathbf{J}^+_{esm}$ は $\mathbf{J}_{esm}$ の擬似逆行列である．

ここで ESM の立式で FC，IC における最小となる解 $\mathbf{x}_o$ を記述すると，FC，IC の順に

$$\mathbf{x}_o = -(\mathbf{J}_{I^*}\mathbf{J}_w\mathbf{J}_H)^+\mathbf{y}(\mathbf{0}) \tag{1.57}$$

$$\mathbf{x}_o = (\mathbf{J}_I\mathbf{J}_w\mathbf{J}_H)^+\mathbf{y}(\mathbf{0}) \tag{1.58}$$

となる．ただし，ホモグラフィの更新規則は FC で $\hat{\mathbf{H}} \leftarrow \hat{\mathbf{H}}\mathbf{H}(\mathbf{x})$，IC で $\hat{\mathbf{H}} \leftarrow \hat{\mathbf{H}}\mathbf{H}(\mathbf{x})^{-1}$ である．

Eq.(1.58) の解を更新する際，特殊線形群の性質である $(w(\mathbf{H}))^{-1} = w(\mathbf{H}^{-1})$ を用いることで，Eq.(1.59) のように IC 型から FC 型の更新規則に書き換えることができる．

$$\hat{\mathbf{H}}\mathbf{H}(\mathbf{x}_o)^{-1} = \hat{\mathbf{H}}\mathbf{H}(-\mathbf{x}_o) \tag{1.59}$$

したがって，$\mathbf{J}_{esm}$ は FC と IC のヤコビ行列の平均を取る形となる．これは，ESM の解がテンプレート側から近づく解と現在推定されている画像側から近づく解の平均を取っていることを意味している．

$$\mathbf{J}_{esm} = \frac{1}{2}(\mathbf{J}_{FC} + \mathbf{J}_{IC}) \tag{1.60}$$

1.4.6 照明変化への対処

前節で述べた手法は，追跡対象の平面パターンが微小変形したとき，輝度の差の二乗和を最小にする二次元射影変換のパラメータを計算していた．そのため弱いノイズがあった場合でもトラッキングが可能であるが，局所的な光の反射などには非常に影響を受けやす

く，誤った領域の推定しまう問題点があった．この節では，どのように光の影響を ESM の計算に適用するかを述べる．

光の反射特性のモデルは，拡散反射 $\alpha_i I_d(\mathbf{p}_i)$，鏡面反射 $\eta_i I_s(\mathbf{p}_i)$，環境光の変化，そしてカメラのバイアス β によって決定され, それは Eq.(1.61) とおける．

$$I^{'}(\mathbf{p}_i;\alpha_i,\eta_i,\beta) = \alpha_i I_d(\mathbf{p}_i) + \eta_i I_s(\mathbf{p}_i) + \beta \tag{1.61}$$

ここで任意の物体を追跡する際，拡散反射や鏡面反射のパラメータは未知である為，Malis ら [57] は拡散反射と鏡面反射の項を一つにまとめた Eq.(1.62) を用いて推定を行っている．

$$I^{'} = \gamma I + \beta \tag{1.62}$$

ここで，γ が拡散反射と鏡面反射をまとめたパラメータである．さらに，平面パターンは一様な光の影響を受けているとは限らない為，追跡領域のピクセルを数分割し，それぞれの領域に異なる γ を置いて推定を行っている．

画像を n 分割した場合，ESM で推定すべきパラメータ数は，ホモグラフィの 8，画像分割分の γ の n，そして環境光とカメラバイアスの影響を考慮した β の 1，合計して n+9 である．このパラメータを $\theta = [\tilde{x}^T, \tilde{\gamma}^T, \tilde{\beta}]^T$ とする．分割した j 番目のブロックにおける i 番目のピクセルの輝度を

$$I^{'}(w(p^*_{ij};H);\gamma_j,\beta) = \gamma_j I(w(p^*_{ij};H)) + \beta \tag{1.63}$$

と定義すると，前節の Eq.(1.45) は以下のように書き換えることができる．

$$f(\theta) = \frac{1}{2}\|I^{'}_{ij} - I^*_{ij}\|^2 = \frac{1}{2}\|d(\theta)\|^2 \tag{1.64}$$

Eq.(1.64) を ESM の手法で線形化すると，j 番目のブロックにおける計算は

$$\frac{1}{2}\begin{bmatrix} \gamma_j(\mathbf{J}_I + \mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_\mathbf{H} & I_{1j} + I^*_{1j} & 2 \\ \gamma_j(\mathbf{J}_I + \mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_\mathbf{H} & I_{2j} + I^*_{2j} & 2 \\ \vdots & & \\ \gamma_j(\mathbf{J}_I + \mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_\mathbf{H} & I_{kj} + I^*_{kj} & 2 \end{bmatrix} \begin{pmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_8 \\ \tilde{\gamma}_j \\ \tilde{\beta} \end{pmatrix} = - \begin{pmatrix} I^{'}_{1j} - I^*_{1j} \\ I^{'}_{2j} - I^*_{2j} \\ \vdots \\ I^{'}_{kj} - I^*_{kj} \end{pmatrix} \tag{1.65}$$

となる．全てのブロックを考慮した式は，k をブロック内の画素数とすると

$$\frac{1}{2}\begin{bmatrix}
\gamma_1(\mathbf{J}_I+\mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_\mathbf{H} & I_{11}+I_{11}^* & 0 & \cdots & 0 & 2 \\
\gamma_1(\mathbf{J}_I+\mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_\mathbf{H} & I_{2j}+I_{21}^* & 0 & \cdots & 0 & 2 \\
\vdots & & & & & \\
\gamma_1(\mathbf{J}_I+\mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_\mathbf{H} & I_{k1}+I_{k1}^* & 0 & \cdots & 0 & 2 \\
\gamma_2(\mathbf{J}_I+\mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_\mathbf{H} & 0 & I_{12}+I_{12}^* & \cdots & 0 & 2 \\
\gamma_2(\mathbf{J}_I+\mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_\mathbf{H} & 0 & I_{22}+I_{22}^* & \cdots & 0 & 2 \\
\vdots & & & & & \\
\gamma_2(\mathbf{J}_I+\mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_\mathbf{H} & 0 & I_{k2}+I_{k2}^* & \cdots & 0 & 2 \\
\vdots & & & & & \\
\vdots & & & & & \\
\gamma_n(\mathbf{J}_I+\mathbf{J}_{I^*})\mathbf{J}_w\mathbf{J}_\mathbf{H} & 0 & 0 & \cdots & I_{kn}+I_{kn}^* & 2
\end{bmatrix}\begin{pmatrix}\tilde{x}_1\\ \vdots\\ \tilde{x}_8\\ \tilde{\gamma}_1\\ \vdots\\ \tilde{\gamma}_n\\ \tilde{\beta}\end{pmatrix} = -d(0) \tag{1.66}$$

の形で表され，これを解くことでパラメータを推定することができる．パラメータ x は前節と同じように更新され，γ,β は Eq.(1.67) のように一つ前の推定値に加算することで更新される（ここで γ の初期値は 1，β の初期値は 0 である）．

$$\begin{aligned}\hat{\gamma}_j &\leftarrow \hat{\gamma}_j+\tilde{\gamma}_j\\ \hat{\beta} &\leftarrow \hat{\beta}+\tilde{\beta}\end{aligned} \tag{1.67}$$

1.4.7 モーションブラーへの対処

本研究ともっとも関連が深いのが，モーションブラーへの対処を述べた研究 [31, 44] である．Jin ら [31]（および Mei ら [44]）のアイデアは，モーションブラーの影響を取り除くには，効果および計算効率の観点から，撮影画像のブラーを何らかの方法で除去しようとするのではなく，テンプレートの方にブラーを反映させるというものである．

モーションブラーが，画像上で均一な $\mathbf{v}(=[v_x,v_y]^\top)$ 方向への等速ブラーであるとすると，モーションブラーを含む画像は，瞬時画像（ブラーのない画像）$\mathcal{I}^u(\mathbf{p})$ を用いて，$\mathcal{I}(\mathbf{p})=\int_0^1 \mathcal{I}^u(\mathbf{p}-\mathbf{v}t)dt$ によって与えられる．さらに Mei らは，任意の等速ブラーが

$$\mathcal{I}_{\hat{\mathcal{H}},\mathcal{H}(\mathbf{x})}(\mathbf{p}) = \int \mathcal{I}^u(\mathrm{e}^{-t\hat{\mathcal{H}}}\mathrm{e}^{-t\mathcal{H}(\mathbf{x})}\mathbf{p})dt \tag{1.68}$$

と表現できることを示した．ここで $\mathcal{H}$ は，それに対応する射影変換 $\mathbf{H}$ を与える Eq.(1.28) の，指数関数の内部の行列を表す．また $\hat{\mathcal{H}}$ は，$\mathcal{H}(\mathbf{x})$ の累積（現フレームでシャッターが

開いてからの積分）を表す．モーションブラーが完全に平面の運動によって決まるとすると，Eq.(1.29) は

$$\mathbf{x}_o = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \left[\mathcal{I}(\hat{\mathbf{H}}\mathbf{H}(\mathbf{x})\mathbf{p}) - \mathcal{I}^*_{\hat{\mathcal{H}}_b,\mathcal{H}(\mathbf{x})}(\mathbf{p}) \right]^2 \tag{1.69}$$

に置き換えられる．（ここでは式を簡素化するため，フレーム間隔中シャッターが開き続けるものとした．詳しくは [44] を参照．）

なお，Eq.(1.69) にしたがって計算を行えば原理的にはモーションブラーを扱えることになる．しかしながらわれわれの経験では，[44] で報告されているような効果は一定程度認められるものの，モーションブラーをモデル化しない場合に比べて反復計算の収束性が低下し，追跡の安定性が相当程度損なわれるようである．このことは，Eq.(1.69) において，差分をとる 2 つの成分がともに未知数 $\mathbf{x}$ に依存し，関数がより複雑な構造を持つためであると思われる．

1.4.8 画像類似度の分類

本章では，画像の類似度を SSD で統一して説明したが，このほかにもいくつか画像類似度の取り方があり，それらは階層的な構造を持っている．

SAD(Sum of Absolute Difference)，SSD(Sum of Squared Difference) は，濃淡変化を許容しない類似度であり以下の式で計算できる．

$$f_{SAD} = \sum_{i=0} |I(\mathbf{p}_i) - I^*(\mathbf{p}^*_i)|. \tag{1.70}$$

$$f_{SSD} = \sum_{i=0} (I(\mathbf{p}_i) - I^*(\mathbf{p}^*_i))^2. \tag{1.71}$$

これらは，値が小さいほど画像間が類似していると考えている．SSD の場合，誤差が二乗で効いてくるため，SSD に比べて誤差に敏感である．

NCC(Normalized Cross-Correlation)，ZNCC(Zero-mean Normalized Cross-Correlation) は，濃淡のスケール変化を許容する類似度であり以下の式で計算できる．

$$f_{NCC} = \frac{\sum_{i=0} I(\mathbf{p}_i) I^*(\mathbf{p}^*_i)}{\sqrt{\sum_{i=0} I(\mathbf{p}_i)^2 \times I^*(\mathbf{p}^*_i)^2}}. \tag{1.72}$$

$$f_{ZNCC} = \frac{\sum_{i=0} (I(\mathbf{p}_i) - I_{mean})(I^*(\mathbf{p}^*_i) - I^*_{mean})}{\sqrt{\sum_{i=0} (I(\mathbf{p}_i) - I_{mean})^2 \times (I^*(\mathbf{p}^*_i) - I^*_{mean})^2}}. \tag{1.73}$$

これらは，画像間のコサイン距離を計算しているため，1 に近くなるほど類似していることになる．コサイン距離はスケールに不変であるため，画像の濃淡変化に比較的強い．ZNCC で濃淡の平均値を引いて計算する理由は，画像全体にスケール倍でない照明変動が起こった時，類似度の値のゆらぎが大きくなる可能性があるからである．

MI(Mutual information) は，比較する画像の濃淡値が一対一に写像されているならば，類似度は最大になる．つまり，テンプレートと入力画像の対応するピクセルの濃淡値が全単射の関係にあれば良い．画像ヒストグラムで考えると，ヒストグラムのビンを入れ替えたような画像，例えば，濃淡を反転したことに相当する画像でも正しく類似度を計算できる．この類似度は以下の式で計算できる．

$$f_{MI} = H(I) + H(I^*) - H(I, I^*). \tag{1.74}$$

$H(I)$ は画像 I のヒストグラムのエントロピーであり，$H(I, I^*)$ は画像 I と I^* の同時確率分布のエントロピーである．

POC(Phase Only Correlation) は，画像の振幅情報は使わず，位相だけに限定して相関を取り，類似度を計算している [29]．画像の位相情報，つまりは輪郭などのエッジの位置情報のみの類似度であるため，比較する画像の濃淡値が全単射の関係を満たしていなくても，形状が一致さえすれば類似度が最大になる．式は以下の通りである．

$$f_{POC} = \mathcal{F}^{-1}\left[\frac{F(k_1, k_2)\overline{G(k_1, k_2)}}{|F(k_1, k_2)\overline{G(k_1, k_2)}|}\right]. \tag{1.75}$$

ここで $F(k_1, k_2)$ および $G(k_1, k_2)$ は I^* および I を二次元離散フーリエ変換したものである．

このように，類似度の取り方で適用できる問題が異なっているため，用途に応じて方法を選ばなくてはならない．

以降の節では，はじめに特徴点に基づく SfM の基礎となっている知識の紹介を行う．加えて，直接法の SfM における最も単純なケースである平面追跡の従来研究について紹介し，その理論について系統立てて説明する．

1.5 本論文の構成

本論文の構成は以下の通りである．第 2 章では，特徴点に基づく方法において，ローリングシャッターに対処するための計算モデルについて紹介し，実環境の実験で提案手法の有効性を示す．第 3 章では，直接法を用いた平面追跡手法において，平面が傾いたり，遠くに離れてしまう際に発生する，追跡領域の実効的な解像度低下の問題について指摘し，

そのモデル化について述べ，実環境の実験で提案手法の有効性を示す．さらに，単一平面追跡から複数平面の追跡に拡張する方法についても説明していく．第 4 章では，提案した直接法の点群と直線追跡の混合モデルについて紹介する．まずはじめに，基本原理となる直接法における点追跡モデルの説明を行い，それから直線モデルに拡張するための定式化を示す．実画像を使った実験によって，点群のみを追跡した場合に比べて提案手法のほうが安定性と精度の面で優れることを実証する．第 5 章では，研究の成果や今後の展望について述べる．付録では，直接法の要素技術である平面追跡手法において，計算処理を高速化するための GPU 実装について述べる．

第2章　特徴点に基づくSfMのためのローリングシャッター歪みの克服

この章では，特徴点ベースのSfMにおけるローリングシャッター(RS)の問題に焦点を当て，critical motion sequence(CMS)について考える．カメラが純回転してるRSモデルを用いて，RS歪みが仮想的に置いたカメラの2つの内部パラメータとレンズ歪みに類似した1つのパラメータの計3つで近似できることを示す．そして，これを仮想カメラの自己校正の問題として再定式化する．その際にスキューとアスペクト比は未知であり、画像シーケンスにおいて変化するようなものを想定している．この導出過程においてCMSの一般表現について説明する．さらに，提案した手法が最近の文献で報告されたCMSによる縮退現象をうまく説明できることを示し，その対処方法について述べる．我々の理論は実験結果ともよく一致しており，通常のバンドル調整で起こる縮退現象を提案手法によって解決することができる．

2.1　RS SfMの研究背景

スマートフォンやアクションカメラのような一般的に普及している画像デバイスはCMOSセンサーを使っており，その大部分がローリングシャッター方式を採用している．ここ数年，ローリングシャッター（RS）歪みをもつ複数視点の画像を使ったSfMに対する関心が増えつつあり，様々な異なる問題設定でRS歪みに対処する方法が提案されてきた[19, 3, 24, 5]．最も一般的な問題解決の仕方は，ローリングシャッターに対応したバンドル調整(RSBA)を用いる方法であり，これはRS歪みを仮定した上で，バンドル調整の枠組みを使い通常のカメラパラメータと一緒に歪みのパラメータを最適化するものである[24, 23, 55]．

多くのRS SfMの方法が研究されてきたにもかかわらず，縮退条件に関する研究はあまりなされておらず，唯一例外として，Albl ら[6]の研究があるのみである．彼らは，RS SfMする際に縮退するケースが存在していること示し，どのようなメカニズムでそれが起こるのか説明している．しかし，この報告では縮退の特殊な1ケースしか取り扱っていない．そのため，もっと多くの縮退条件が存在しているのではないかなどといった疑問が

まだ残っている．

本研究の背景にある最も大きな関心は，これらの疑問に答えることであり，RS SfM の一般的な縮退条件を明らかにすることである．しかし，RS SfM は多変数の非線形問題であるため，これを解決するのは容易ではない．この難しさを緩和するために，この研究では問題を実用的な範囲の RS 歪みに絞って簡略化し，過去に膨大な CMS の研究がなされてきたカメラの自己校正と類似した形で解決を図っていく．具体的にはまずはじめに，RS 歪みがカメラの位置に依存しない等速度のカメラ回転によって発生しているものと仮定した．さらに，そのカメラ回転が小さく線形化できると仮定し，RS 歪みが仮想的に置いたカメラの内部パラメータ（スキューとアスペクト）の 2 つと，レンズ歪みに類似した 1 つの非線形の変換パラメータで表現できることを示す．このようにモデルを仮定することによって，RS SfM を画像中でスキューとアスペクトが変化するような仮想的なカメラの自己校正問題に置き換えることができる．この自己校正の計算過程から，推定結果が縮退するような CMS の一般表現について導出をする．そして，[6] の研究で得られたものと一致する CMS の表現を説明し，それに対処する新しい解決策を提案する．最終的に，一連の実験結果を通して提案手法の有用性と有効性を示す．

2.2 関連研究

単一または多視点画像における RS 歪の問題は，これまで様々な研究がおこなわれてきた．既存研究は，問題の種類で分類すると，絶対姿勢 [3, 40, 5, 54]，相対姿勢 [12]，複数視点の最適化（BA）[23, 24, 55]，ステレオ視 [4, 53]，そして画像補正や安定化 [51, 30, 21] に分類される．RS 歪を誘引するカメラの運動にかかる条件には色々あり，視点間でカメラ運動に相関があるかどうか，歪を誘引するカメラ運動が視点間で独立か連続かどうかなどが挙げられる．

本研究に最も関連が近いのは，Albl ら [6] の研究である．彼らの研究では，まずはじめに RS SfM において縮退が発生し得ることを指摘し，そして，一般的に使われる RS モデル（画像上でカメラが線形に回転して，等速に平行移動する）において CMS が存在することを示した．ここで示された CMS は，全ての画像が空間上で同じ y 方向（つまりは，RS の読み取り方向）で撮影された場合としており，そのような画像集合で復元結果される RS カメラの位置とシーン形状が全て同一の平面上に乗っているような状況が考えられる（つまりは縮退している）ことを証明をしている．

本研究では上記の研究とは異なり，カメラが純回転している仮定を置いた RS カメラモデルで CMS の一般表現を導出するが，上で示されたような CMS について違う方法で説

明することができる．そして，その問題の対処法についても解説していく．もう一つ異なる点は，この導出には線形化によって近似された回転行列をさらに近似する必要があるということである．そのため，厳密にいうとこの近似が有効な範囲に限り，RS SfM の縮退が同様に起こり得ると言える．しかし，ほとんどの条件でこの近似は問題なく，のちに示す実験結果ともよく一致している．

先ほど述べたように，提案手法の核となるアイディアはカメラの自己校正の問題に置き換えて取り扱うことができる．カメラの自己校正は，1990 年代から 2000 年代初頭にかけて広く研究されてきた [42, 26, 50, 25, 27]．それらの研究では，多視点画像において各カメラの内部パラメータが全て未知の場合に，SfM における 3 次元復元の射影的不定性の問題を解決できないことを示している．そして，スキューが 0，またはどれかひとつの内部パラメータが画像間で変わらない場合に射影的不定性を取り払うことができ，自己校正が可能になるとしている．しかし，これらはカメラモーションが普通の場合のみを想定し，計算可能かどうかを保証している．そのため，カメラモーションが CMS の場合は縮退を起こし，自己校正はできなくなってしまう．異なる設定の自己校正問題はそれぞれ CMS の条件が異なっており，実用的に重要な一部の設定のみがよく研究されている [63, 33]．焦点距離だけが不明でカメラごとに変化する場合については [64] が挙げられる．本研究では，スキューとアスペクトだけが不明でカメラ間で変化するような状況を考える．この設定は，一般的な問題ではあまり利用価値がなかったためこれまで研究されておらず，おそらく自己校正の研究題材として初めての試みになるであろう．

2.3 ローリングシャッターのモデル

2.3.1 定速度運動モデル

$\mathbf{X} \equiv [X, Y, Z]^\top$ と $\mathbf{x} \equiv [x, y, z]^\top$ をそれぞれワールド座標とカメラ座標とする．これら二つの座標変換は次のように与えられる．

$$\mathbf{x} = \mathbf{R}(\mathbf{X} - \mathbf{p}), \tag{2.1}$$

$\mathbf{R}$ は回転行列，$\mathbf{p}$ はワールド座標におけるカメラの平行移動ベクトルである．画像が撮影されている間に，カメラが等速に動いているとすると，ローリングシャッター (RS) 歪み

は以下のようにモデル化される．

$$\begin{bmatrix} c \\ r \\ 1 \end{bmatrix} \propto \mathbf{R}(r\boldsymbol{\phi})\mathbf{R}\{\mathbf{X} - (\mathbf{p} + r\mathbf{v})\}, \tag{2.2}$$

ここで c と r はそれぞれ，列 (x) と行 (y) の座標になっており，カメラシャッターは r の小さい方から大きい方へ向かって閉じている．$\mathbf{R}(r\boldsymbol{\phi})$ と $\mathbf{v}$ はカメラモーションの回転行列と平行移動ベクトルで，$\boldsymbol{\phi} = [\phi_1, \phi_2, \phi_3]^\top$ は回転軸角度表現になっている．ここで注意なのが，$\mathbf{R}$ と $\mathbf{p}$ はシャッターが $r = 0$ で切られたときのカメラポーズになっている．

角度 $\boldsymbol{\phi}$ が小さいと仮定すると，Eq.(2.2) を以下のように近似できる．

$$\begin{bmatrix} c \\ r \\ 1 \end{bmatrix} \propto (\mathbf{I} + r[\boldsymbol{\phi}]_\times)\mathbf{R}\{\mathbf{X} - (\mathbf{p} + r\mathbf{v})\}, \tag{2.3}$$

ここで

$$[\boldsymbol{\phi}]_\times = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}. \tag{2.4}$$

このモデルは，画像撮影時のカメラの平行移動成分を表現する方法が過去の研究 (e.g.,[5]) と少し違っている．我々のモデルでは，カメラの空間位置を回転成分と分けてパラメータ化しており，こちらのほうがより自然な表記であるといえる．

2.3.2 回転のみの運動モデル

ここからは，$\mathbf{v}$ が小さく無視できる場合を考える．これは一般的に，カメラがシーン上の物体に近くなければ十分に良い近似となる．$\mathbf{v} = \mathbf{0}$ と置き，Eq.(2.1) で $\mathbf{x} \equiv [x, y, z]^\top$ とすると, Eq.(2.3) は以下のように書き換えられる．

$$\begin{bmatrix} c \\ r \\ 1 \end{bmatrix} \propto (\mathbf{I} + r[\boldsymbol{\phi}]_\times) \begin{bmatrix} x \\ y \\ z \end{bmatrix} \propto (\mathbf{I} + r[\boldsymbol{\phi}]_\times) \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \tag{2.5}$$

ここで，$x' \equiv x/z$ ，$y' \equiv y/z$ である．

2.3.3 アフィンカメラ近似に基づく2段階モデル

Eq.(2.5) はパースペクティブカメラが一定の速度で回転した時に発生する RS 歪みのモデルである．ここで，アフィンカメラのモデルとして近似すると

$$\begin{bmatrix} c \\ r \\ 1 \end{bmatrix} \propto (\mathbf{I} + r[\boldsymbol{\phi}]_\times) \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \approx \begin{bmatrix} x' - r\phi_3 y' + r\phi_2 \\ r\phi_3 x' + y' - r\phi_1 \\ 1 \end{bmatrix}. \tag{2.6}$$

となる．これは RS 歪みの主成分を抽出したモデルになっており，x', y', r が小さいときに成り立つ近似である．この近似は，一般的なカメラの場合は画像中心周りの点で当てはまり，望遠レンズの場合は任意の点で当てはまる．ここで注意しておきたいのが，このアフィンカメラモデルは SfM の投影モデルには使っておらず，Eq.(2.12) のように通常のパースペクティブカメラモデルを使用している．したがって上記の近似は，RS 歪みモデルで計算される画像座標の 2 次の項の影響を単に無視するためだけにつかっている．この近似の妥当性に関しては 2.5 で示す．

Eq.(2.6) は $f : [x', y'] \mapsto [c, r]$ の変換を与えており，この f は 2 つの変換によって近似することができる．

Proposition 1. ϕ_1, ϕ_2, ϕ_3 が小さい時, $f : [x', y'] \mapsto [c, r]$ は以下のように近似できる．

$$f \approx f_p \circ f_d, \tag{2.7}$$

ここで $f_d : [x', y'] \mapsto [x'', y'']$ の定義は

$$x'' = x' - \phi_3 y'^2, \tag{2.8a}$$

$$y'' = y' + \phi_3 x' y', \tag{2.8b}$$

であり，$f_p : [x'', y''] \mapsto [c, r]$ は

$$\begin{bmatrix} c \\ r \\ 1 \end{bmatrix} \propto \begin{bmatrix} 1 & \phi_2 & 0 \\ 0 & 1 - \phi_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix}. \tag{2.9}$$

Proof. (2.7) の近似が正しいことを，ϕ_1, ϕ_2, ϕ_3 の 1 次近似用い，左辺と右辺が等しいことで示す．右辺において，Eqs.(2.8) を Eq.(2.9) に代入し，ϕ_1,ϕ_2, ϕ_3 の 2 次の項を無視することで

$$c = x' + \phi_2 y' - \phi_3 y'^2, \tag{2.10a}$$

$$r = y' - \phi_1 y' + \phi_3 x' y'. \tag{2.10b}$$

となる．左辺において，Eq.(2.6) の右式の第二要素である $r = r\phi_3 x' + y' - r\phi_1$ を第一要素に代入して 2 次の項を無視すると (2.10a) となり，同じ手順で第二要素の r に自身の r を代入し，2 次の項を無視すると (2.10b) となる． □

2.4 RS カメラの自己校正

RS カメラモデルとして，Eqs.(2.7)-(2.9) を用いる．まずはじめに，自己校正の問題としてモデルを定式化した SfM について示す．

2.4.1 問題の定式化

Proposition 2. 内部パラメータが既知のカメラによって撮影された画像から *SfM* することを想定する．それぞれの画像でパラメータが未知の状態の *RS* カメラモデルが *Eqs.(2.7)-(2.9)* で与えられると仮定すると，*SfM* の問題は，画像間でスキューとアスペクトと特別なレンズ歪みが変化する仮想的なカメラの自己校正の問題に等しくなる．

Proof. $f = f_p \circ f_d$ で与えられる 2 ステップモデルの最初の要素である f_d はレンズ歪みと解釈することができる．最もよく用いられるレンズ歪みのモデルは以下である．

$$x'' = x' + x'(k_1\lambda^2 + k_2\lambda^4) + 2p_1x'y' + p_2(\lambda^2 + 2x'^2) \tag{2.11a}$$

$$y'' = y' + y'(k_1\lambda^2 + k_2\lambda^4) + p_1(\lambda^2 + 2y'^2) + 2p_2x'y', \tag{2.11b}$$

ここで $x' \equiv x/z$, $y' \equiv y/z$, $\lambda^2 \equiv x'^2 + y'^2$ である．右辺の第 2 項は radial distortion(放射状の歪み) であり，第 3 項は tangential distortion(円周方向の歪み) である．Eq.(2.8) の変換 f_d は同様の形をしており，2 次の項である y'^2 と $x'y'$ は tangential distortion の項に現れている．

2 つめの要素である Eq.(2.9) の f_p について考える．行と列の座標 $[c, r]$ は，カメラの内部パラメータ $\mathbf{K}$ によって画像座標 $[u, v]$ に以下のように変換される．

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \mathbf{K} \begin{bmatrix} c \\ r \\ 1 \end{bmatrix} \propto \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c \\ r \\ 1 \end{bmatrix}, \tag{2.12}$$

ここで f と (u_0, v_0) はカメラの焦点距離と画像中心となっている．問題を簡潔化するため，スキューを 0，アスペクトを 1 と置く．この値を変えてもこれからの説明内容には影響は

ない．上の式に Eq.(2.9) を代入すると

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \phi_2 & 0 \\ 0 & 1-\phi_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} f & \phi_2 f & u_0 \\ 0 & (1-\phi_1)f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix}.$$

となる．これは内部パラメータとして ϕ_1 (厳密には $1-\phi_1$) のアスペクトを持ち，ϕ_2 のスキューを持つ仮想的なカメラ行列と考えることができる． □

2.4.2 自己校正の理論

このようにわれわれの RS モデルを使うと，RS 歪下の SfM はある自己校正の問題として定式化できる．自己校正とは，同一シーンを撮影した画像のみから，多視点幾何に基づいて，カメラの（部分的に）未知の内部パラメータを推定し，SfM の metric reconstruction を得ることを言う．これは 1990 年代から 2000 年始めにかけてよく研究された問題である [42, 26, 50, 25, 27, 22]．それら研究成果を以下で要約しつつ，われわれのケースとの関係を述べる．

自己校正の Feasibility

自己校正は 2000 年前後によく研究され，一定の結論を見た．全内部パラメータが未知の場合，射影復元しか得られない．自己校正には，それらについての知識を必要とする．当初は全画像の内部パラメータが未知だが共通の場合が議論され [42]，その後関心は一部既知の場合に移った．スキューとアスペクトのみが既知で残りは未知で可変の場合 [26] が調べられ，スキューが 0 であれば十分 [50, 25] と分かり，最終的には 5 パラメータのうち 1 つだけコンスタントなら十分で，あとは可変で未知で構わない [27] と分かった．したがってわれわれのケース（スキューとアスペクトが視点ごとに異なり未知，それ以外は既知）でも，この理論に従えば自己校正は可能である．

このように理論的には，わずかなパラメータの知識がありさえすれば自己校正は実現可能であるということになるが，現実はその通りではない．理由は 2 つあり，1 つは，カメラの姿勢変動が一般的でないといけないという，後述する CMS の存在である．もう一つは，あまりに多くの未知パラメータを精度よく定めるのは難しいことである．後者の理由で，第一に焦点距離のみ [50] を，場合によっては principal point を一緒に [33] 推定する場合が実用上は多い．本研究の場合，既知と未知のパラメータの問題設定が完全に逆転しており，非常に興味深いケースになっている．

加えて，自己校正の問題では満たさなければならないカウント条件も存在している．m をカメラの数，n_k を既知である内部パラメータの数，そして n_f を未知の内部パラメータの数すると．必須条件 [22] は以下のようになる．

$$mn_k + (m-1)n_f \geq 8. \tag{2.13}$$

本研究では，$n_k = 3$ and $n_f = 0$ であるため，$m \geq 3$ で条件が満たされる．

Critical motion sequence

前節の問題とは別に，理論的に定まるはずの場合でも，カメラの運動（各視点での姿勢）によって問題が縮退し，解が定まらない場合があることが知られる．そのようなカメラ運動のことを CMS(critical motion sequence() と呼ぶ．

CMS の研究は，自己校正が可能な条件に関する研究と並行に進んだ．カメラの内部パラメータが一定の場合の CMS は [63] にある．さらに一部のパラメータが変化する場合についての研究がなされた．スキュー 0，アスペクト既知，焦点距離以外既知の条件下では [33] が挙げられる．最もポピュラーな自己校正の設定である，焦点距離以外の内部パラメータは既知で，焦点距離は可変未知である場合の CMS のカタログは [64] にある．

しかし，本研究で考える場合，つまりスキューとアスペクトが未知・可変でそれら以外は既知という条件は極めて珍しく，研究が見当たらない．そこで Sec.2.4.3 で，この条件の CMS を考える．

レンズ歪

焦点距離などを自己校正時（またはバンドル調整をする時），レンズ歪を一緒に推定することは普通に行われる (i.e., Eq.(2.11) の $k_1, \ldots$ と $p_1, \ldots$)．レンズ歪のそれも，広義の内部パラメータと見なせるが，自己校正の理論研究では，伝統的にそれらは 5 つの内部パラメータとは区別されて取り扱われてきた．実際，最近までその独特さは厳密に議論されてこなかった [68]．歪が持つ非線形性（直線を直線に写さない）から，射影変換パラメータとの区別は容易であるし，十分なシーンの点があればレンズ歪を補正する情報は得られるため，レンズ歪の CMS は直感的には問題になりそうにない．本研究でも同様に考え，f_d のパラメータ ϕ_3 は一意に決まるものと仮定する．

2.4.3 RS カメラの自己校正における CMS

焦点距離と画像中心が既知で，スキューとアスペクトが未知かつ視点ごとに変化する場合の CMS を考える．そのためにまず，[22] の Sec.19.2 にある自己校正の方程式を導出し，我々のケースに当てはめていく．

自己校正の方程式

射影復元が $\{\mathbf{P}^i, \mathbf{X}_j\}$ であると想定する．ここで，$\mathbf{P}^i$ は射影行列，$\mathbf{X}_j$ は空間状の点である．$\mathbf{P}^i = [\mathbf{A}^i \mid \mathbf{a}^i]$ とし，カメラ 1 の座標系を $\mathbf{A}^1 = \mathbf{I}$, $\mathbf{a}^1 = \mathbf{0}$ と仮定する．カメラ i の内部パラメータを $\mathbf{K}^i$ と置くと，画像 i の DIAC(dual image of the absolute conic) は $\omega^{*i} = \mathbf{K}^i\mathbf{K}^{i\top}$ で与えられる．射影復元 (projective reconstruction)$\{\mathbf{P}^i, \mathbf{X}_j\}$ から 3 次元形状復元 (metric reconstruction)$\{\mathbf{P}^i\mathbf{H}, \mathbf{H}^{-1}\mathbf{X}_j\}$ に変換する 3 次元射影変換 $\mathbf{H}$ は以下のように置くことができる．

$$\mathbf{H} = \begin{bmatrix} \mathbf{K}^1 & \mathbf{0} \\ -\mathbf{p}^\top\mathbf{K} & 1 \end{bmatrix}. \tag{2.14}$$

そして，自己校正の式 ($i = 1, \ldots$) は

$$\omega^{*i} = \left(\mathbf{A}^i - \mathbf{a}^i\mathbf{p}^\top\right)\omega^{*1}\left(\mathbf{A}^i - \mathbf{a}^i\mathbf{p}^\top\right)^\top. \tag{2.15}$$

となる．ここで ω^{*i} ($i = 1, \ldots$) と $\mathbf{p}$ は未知のパラメータである．内部パラメータに関するいくつか分かっていれば，ω^{*i} の各要素に拘束を与えることができる．

これから，我々の考えるケースでこの拘束を導くことにする．画像中心は全ての画像で既知であるため，それらを画像上で $(0, 0)$ の位置に変換することができる．するとカメラの内部パラメータは

$$\mathbf{K}^i = \begin{bmatrix} f^i & s^i f^i & 0 \\ 0 & \alpha^i f^i & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.16}$$

となり，DIAC は

$$\omega^{*i} = \mathbf{K}^i\mathbf{K}^{i\top} = \begin{bmatrix} f^2 + s^2 f^2 & s\alpha f^2 & 0 \\ s\alpha f^2 & \alpha^2 f^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.17}$$

となる．ここでは，行列内の添え字 i は簡略化のため除外した．この式から各カメラの DIAC の要素を拘束できる．左上の 2×2 行列を使って s と α を消去すると

$$\omega^{*i}_{11}\omega^{*i}_{22} = \omega^{*i}_{22} f^2 + \omega^{*i2}_{12}, \tag{2.18a}$$

が得られる．(1, 3) と (2, 3) 要素は

$$\omega_{13}^{*i} = \omega_{23}^{*i} = 0. \tag{2.18b}$$

となる． Eq.(2.15) を上の式に代入し，ω^{*i} $(i \neq 1)$ の要素を消去すると，ω^{*1} と $\mathbf{p}$ だけが未知の方程式が得られる．これらの未知数を解くことで，$\mathbf{H}$ を決定することができ，3 次元復元 (metric reconstruction) が可能になる．

CMS の表現

CMS とは前述した方程式が縮退するようなカメラモーションの集合である．今回取り扱う問題では，カメラ i ごとに Eq.(2.15) を Eqs.(2.18) に代入した 3 つの方程式が存在している．各カメラの方程式は $\mathbf{A}_i$ と $\mathbf{a}_i$ が異なっているため，それぞれ違うものになっている．しかし，カメラ間で $\{\mathbf{A}_i, \mathbf{a}_i\}$ がある特殊な関係性を持っていた場合に縮退が起こりうる．

この縮退を理論的に解析することによって，CMS の全てのパターンを見つけることができる．実際には，カメラモーションを代入してそれが CMS かどうかを判別すれば良いのだが，直感的にそれがどういうカメラモーションであるか理解することは難しい．この問題は一般的に使われる他の自己校正でも同様のことが言え，それらのケースではいくつか実用的に重要な CMS だけに絞って説明を行なっている．同様に我々のケースでも直感的にわかりやすい CMS の 1 事例（[6] に出てくる CMS と一致）について導出する．

Proposition 3. すべての画像がカメラの y 軸に対して平行に撮影された場合，このカメラモーションは *CMS* である．カメラの平行移動成分については任意の値で良い．

Proof. y 軸に回転する角度 φ を持った回転行列を $\mathbf{R}_y(\varphi)$ と定義する．ワールド座標系で，空間上の 3 次元点 $[X, Y, Z]^\top$ は以下のように変換される．

$$\mathbf{K}\left(\mathbf{R}_y(\varphi)\begin{bmatrix}X\\Y\\Z\end{bmatrix} + \begin{bmatrix}t_X\\t_Y\\t_Z\end{bmatrix}\right) = \begin{bmatrix}f(X\cos\varphi + Z\sin\varphi + t_X) + fs(Y + t_Y)\\ f\alpha(Y + t_Y)\\ -X\sin\varphi + Z\cos\varphi + t_Z\end{bmatrix}, \tag{2.19}$$

$\mathbf{K}$ は Eq.(2.16) でパラメータ化されたものである．ここで Y 座標に k をかけたものと，s と α に $1/k$ をかけたものは，$s(Y + t_Y) = sk^{-1} \cdot k(Y + t_Y)$，$\alpha(Y + t_Y) = \alpha k^{-1} \cdot k(Y + t_Y)$ になるため観測される画像は変化しない．したがって，これらの画像からスケール k を決めることはできず，これは CMS のカメラモーションと言える． □

この CMS は，例えばカメラを常に地面に水平に構え，移動しながら画像を撮影する場合に相当する．このような撮影は良く起こりえる．また，完全にそうでなくてもそれに近い場合にも，その近さに応じてパラメータ推定の精度が悪くなることがあるだろう．

CMS の解決方法

この CMS を回避するために，CMS に対するの対策がいくつか考えられる．例えば 1 つの視点を選びスキューかアスペクト比，つまり ϕ_1, ϕ_2 のいずれかについて値を自己校正の枠組みの外で決定し，指定することである．このように値を 1 つ与えるだけで上述のスケール k が決定できることになり，解の不定性を解決できる．実用的には，RS 歪が発生していない視点を 1 つだけ特定し，その視点について $\phi_1^i = 0$ あるいは $\phi_2^i = 0$（あるいはその両方）とすることである．

2.5 実験結果

提案した自己校正に基づく RS SfM の式は，CMS の条件を調べるだけでなく，実際に RS SfM に応用することができる．RS SfM の縮退問題がどのように起こり，どうやって提案手法で解決できるか，また RS モデルに適用した近似は妥当であるか実証するために，提案手法を RS 歪みのある画像に適用する．そして，通常の BA や RS カメラモデルを組み込んだ BA などの結果と真値の比較を行う．

2.5.1 SfM への RS カメラモデルの適用

提案する RS モデルは，SfM のパイプライン中のバンドル調整に組み込むことができる．その手順は以下の通りである．空間の点 **X** の投影は次のように行う．**X** はまず Eq.(2.1) で世界からカメラ座標 **x** に変換し，その後 Eqs.(2.8)，Eq.(2.13) を順に適用して，画像座標 $[u, v]$ を得る．

カメラの内部パラメータ **K** はすべて既知とする．したがってカメラ 1 台あたり，カメラのポーズ 6 自由度と 3 つの RS 歪みパラメータが未知数となる．これらとシーンの点群の 3 次元座標を対象に，バンドル調整を行う．このバンドル調整は，ceres-solver[20] を使って実装した．なお RS 歪みパラメータは（RS 歪みの性質から全カメラを通じて平均 0，分散も大きくないと考え）3 パラメータすべて，初期値は 0 とした．

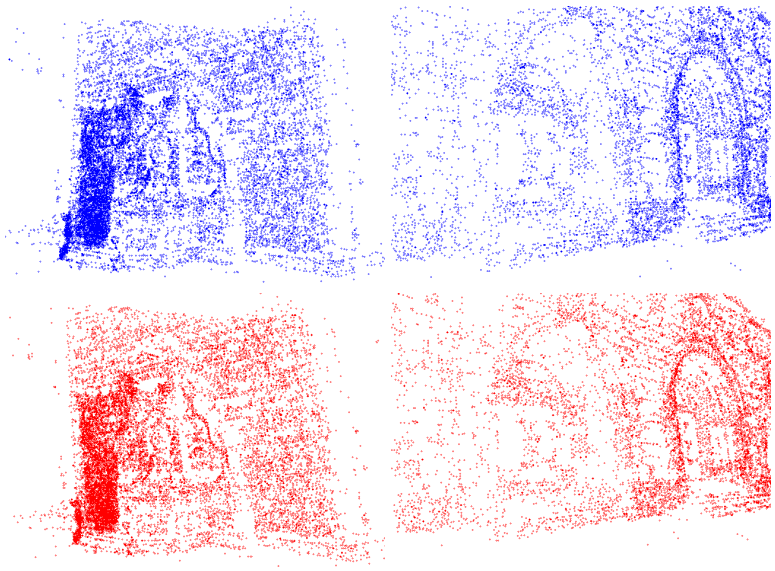


Fig. 2.1: 画像点の例. 青がオリジナル画像．赤が RS 歪みを加えたシミュレーション画像.

Table 2.1: 実験に用いた画像セット.

Sequence	Dataset	# of images
fountain-P11	EPFL[61][1]	11
Herz-Jesu-P8	EPFL	8
castle-P19	EPFL	19
Temple	Middleburry[2]	10(templeR0014-0023)

2.5.2　シミュレーション画像による実験

実験 I：点群の投影に歪みを加える

まず初めに，3 次元復元の結果を真値と比較できるように合成した画像で実験を行なった．この実験では，点群とカメラ姿勢を与えて，これから (2.2) の式を基に RS 歪みのある画像（点群の画像座標）を合成した．Fig.2.1 がその例である．

RS 歪みは SfM の様々な途中経過に影響を及ぼす．例えば，特徴点マッチングやカメラの初期位置の推定，復元される 3 次元形状，バンドル調整などである．提案手法はその最後の処理であるバンドル調整のみに影響があるため，上記の方法ではその有効性を独立に評価することができる．

詳細については以下の通りである．シミュレーション画像をつくるために必要な点群とカメラポーズの計算には Visual SFM [66, 67] を利用した．その際に，2.1 にある公開された様々なデータセットを用いた．

そして，復元された点を RS カメラモデル (2.2) を使ってそれぞれの画像上に投影した．各画像でのカメラモーションはランダムに生成しており，回転行列 $R(r\boldsymbol{\phi})$ に関しては，回転軸 $\boldsymbol{\phi}/|\boldsymbol{\phi}|$ を球面上に一様分布に配置したものを使っており，その回転角速度 $|\boldsymbol{\phi}|$ は $(r_{max}-r_{min})|\boldsymbol{\phi}|$ の値がガウス分布 $N(0,\sigma_{rot}^2)$ に従うように生成したものを使った．ここで r_{max} と r_{min} は画像の上部と下部の行の値になっている．平行移動ベクトルに関しては，それぞれの 3 成分が $N(0,(\sigma_{trans}\bar{t})^2)$ に従うように生成した．ここで，$\bar{t}$ はシーケンス上で連続するカメラ位置の平均移動距離になっている．この実験では，$\sigma_{rot}=0.05$ ラジアン，$\sigma_{trans}=0.05$ に設定し，内部パラメータに関しては，復元の際に用いたものを使用した．さらに，画像上の点 x,y にはガウスノイズ $\varepsilon,\varepsilon' \sim N(0,0.5^2)$ を加えた．

2.1 に対して，以下の手順で 100 回の操作を行なった．画像に与えるノイズとバンドル調整に入力する初期値を毎回生成し，RS バンドル調整を行なった．それぞれの画像にかける RS 歪みに関しては，1 枚目の画像を除いてランダムに生成したものを適用しており，全操作において同じ歪みを使っている．注意点として，1 枚目の画像にはあえて歪みを加えないものとした．

この実験では，上のように生成されたデータに対して 4 つの方法を試した．1 つ目は，RS カメラモデルが入っていない普通のバンドル調整（w/o RS）．2 つ目と 3 つ目は提案したカメラモデルによる RS バンドル調整で，2 つ目のほうは，全ての RS パラメータを最適化したものになっており（w/ RS)，3 つ目のほうは，$\phi_1^1=0$ と固定して，他のパラメータを最適化したものになっている (w/ RS*)．これは Sec.2.4.3 で言及した CMS に対処する手法になっている．4 つ目は，(2.5) の厳密な RS モデルでバンドル調整したものである（w/ RS($r[\phi]_\times$])）．

Fig.2.2 が結果である．このプロットはカメラの回転と平行移動と形状（シーンの点群）の累積誤差ヒストグラムを表している．回転の誤差は全てのカメラで $\mathbf{R}^i\hat{\mathbf{R}}^i$ を計算したものの平均になっている． $\mathbf{R}^i$ と $\hat{\mathbf{R}}^i$ はそれぞれ真値と推定されたカメラ姿勢である．平行移動と形状の誤差については，スケーリングの不定性を消すためにカメラ軌道がもっとも真値に近くなるように相似変換したのち，平行移動は $\mathbf{p}^i, \hat{\mathbf{p}}^i$ の平均誤差で，形状は対応する点との誤差の累計で計算した．

Fig.2.2 から様々なことがわかる．1 つ目は，ϕ_1^1 を固定した “w/ RS*” が全てのデータセットに対して最も良い性能を示したことである．唯一，Herz-Jesu-P8 において “w/ RS” と “w/ RS*” が同等の結果になっている． これは fountain-P11, castle-P19, Temple が CMS の条件に近くなっていることを意味しており，Herz-Jesu-P8 がその条件から離れたものになっている．実際に前者の 3 つはカメラの上下角が小さいため Sec.2.4.3 で議論した CMS のカメラモーションの条件に当てはまっている．一方，後者の Herz-Jesu-P8 では上下角は大き

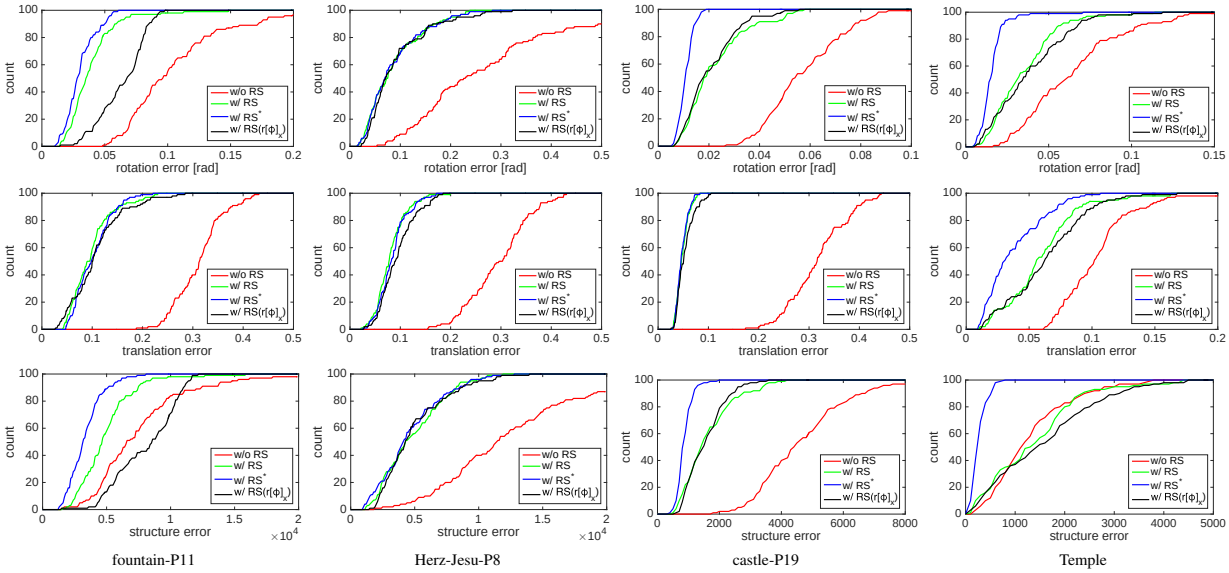


Fig. 2.2: 推定されたカメラの回転・平行移動，復元された形状の累積誤差ヒストグラムの結果．“w/o RS” が通常の BA．“w/ RS” と “w/ RS*” が提案した RS カメラモデルを使った BA (後者のほうは ϕ_1^1 を 0 に固定している)．“w/ RS($r[\phi]_\times$])” がオリジナルの RS モデルを使った BA2.5)．

くなっている．2 つ目は，“w/ RS($r[\phi]_\times$])” が “w/ RS” と似たような振る舞いをしていることである．したがって，提案した RS カメラモデル (Sec.2.3.3) の近似は妥当であったと言える．

4 つのデータセットの典型的な復元結果を Figs.2.3-2.6 に示す．それぞれのデータセットで “w/ RS*” のカメラ軌道や復元された形状の精度が最も良かった．そして “w/ RS” のほう結果では，形状がしばしば縦に伸縮していることがわかる．これは，Fig.2.2 の累積誤差の結果からも同様のことが言え，3 で詳しく述べた CMS でうまく説明することができる．

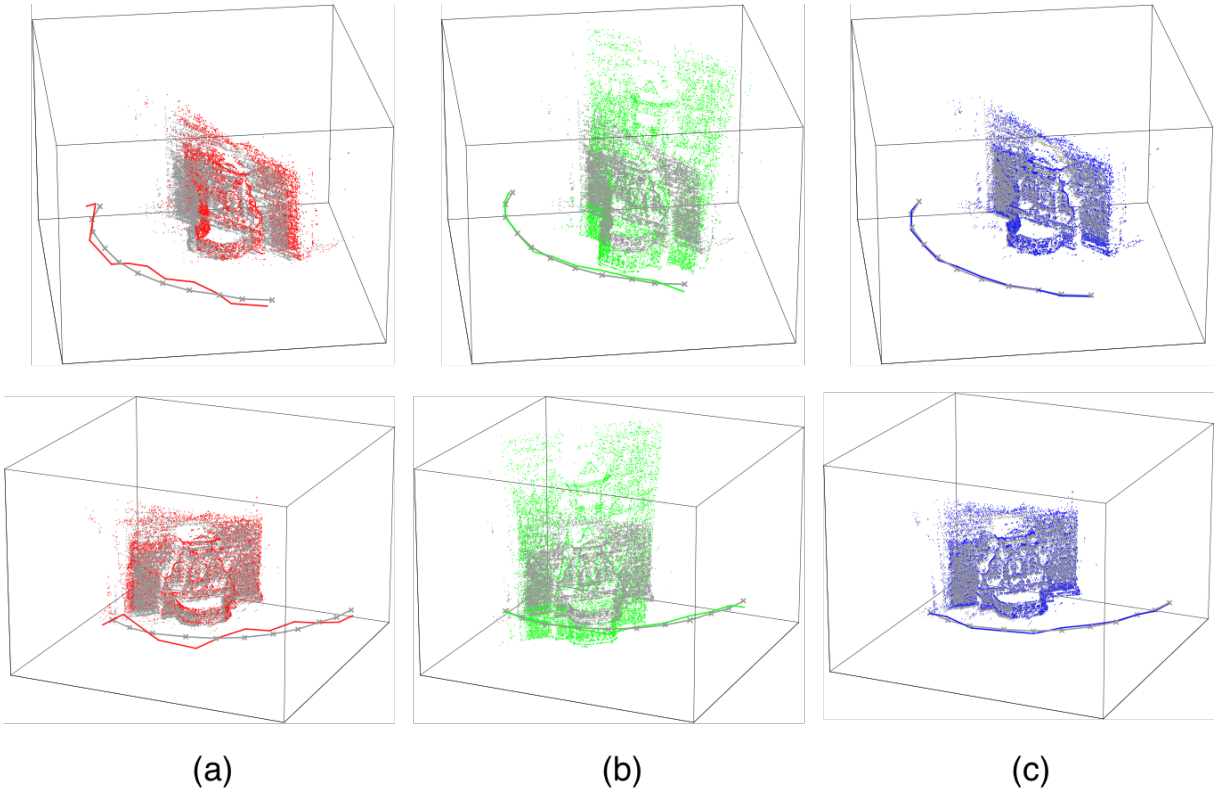


Fig. 2.3: 典型的な復元例：fountain-P11. (a) w/o RS. (b) w/ RS. (c) w/ RS* (ϕ_1^1 fixed). 灰色の点と線はそれぞれシーン形状とカメラ位置である．

実験 II：画像に歪みを加える

実験 I では，RS 歪みの有無にかかわらず，同じ対応点が得られることを前提としている．現実には，RS 歪みは，画像間対応の精度を（それを考慮しないで行った場合は）ならびに BA の初期値の計算精度を劣化させるだろう．この影響が最後の BA を使い物にならなくしてしまうほどであれば，提案手法は意味をなさないことになる．そこでここでは，まずローリングシャッタ歪みの存在しない画像に，カメラ自己回転時のローリングシャッタ歪みを加えた画像を生成する．これら画像を SfM の入力とする，つまり画像間対応を求めることから始め，3 次元復元まで行う．SfM に Visual SFM を用いることで，画像間対応の精度低下も織り込んだ精度評価が可能となる．ただし，実験 I とは違い，RS 歪みにカメラの並進成分は含まれない（奥行き情報を用いた合成が必要となるため）．元となる画像は実験 I と同じものを使用した．生成された画像の例は Fig.2.7 である．

この実験では以下の手順で 100 回の処理を行なった．それぞれの処理で実験 I と同様にランダムな RS 歪みを与え，Visual SFM で 3 次元復元をし，それを “w/o RS” とした．RS 歪みのせいで特徴点マッチングに誤差が出てるが，特に外れ値外れ値処理は行なっていな

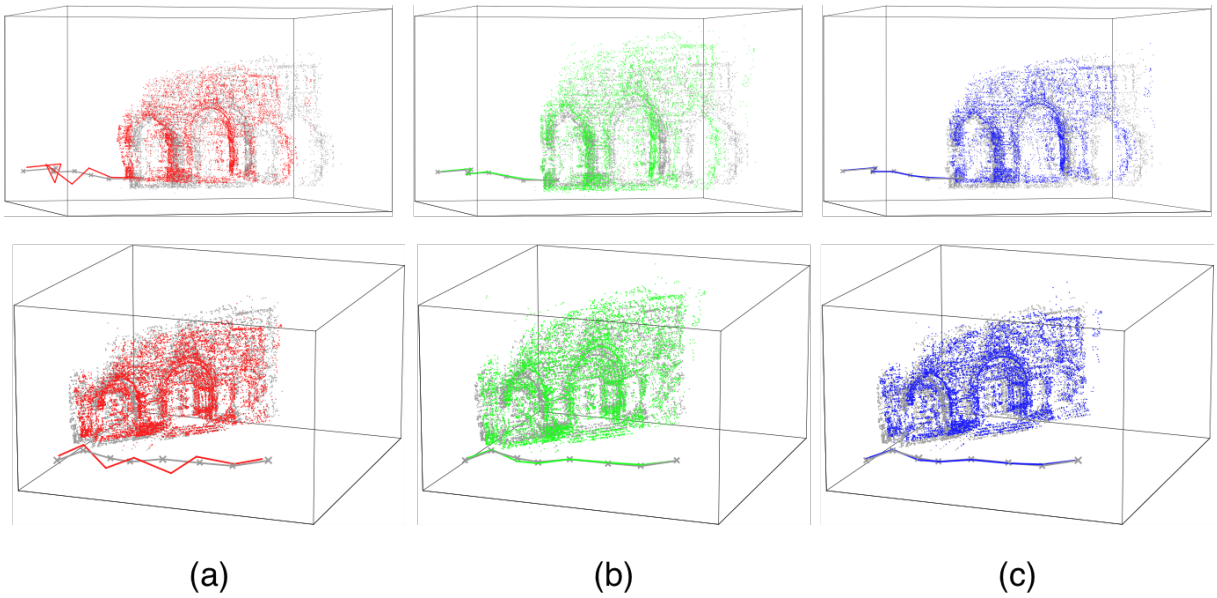


Fig. 2.4: 典型的な復元例：Herz-Jesu-P8. (a) w/o RS. (b) w/ RS. (c) w/ RS* (ϕ_1^1 fixed).

い．“w/o RS” で得られた復元結果を初期値として実験 I と同様に 3 つの RS カメラモデルを適用した．最終的に，歪みなしの画像で得られた結果を真値としてそれぞれの比較を行なった．使用した画像は実験 I と同様である．Fig.2.8 に回転と平行移動の誤差ヒストグラムを，Fig.2.9 に復元された結果を示す．基本的には，これらの結果は実験 I で得られたものと同様になった

2.5.3　実画像による実験

スマートフォン（iPhone5s）のカメラを使い，手持ちで，シーン中を何回か視点を変えて静止画像を撮影する．撮影の際，普通にカメラを静止させて撮影する他，故意にカメラを任意の方向に姿勢変化させつつ画像を撮影する．この 2 度の撮影間で，カメラの向き（RS 歪ありの場合は基準向き）は制御不可能だが，カメラの位置については，大きく変化はしない．あくまで手持ち撮影とは言え，多く見積もって 20cm も変動はない．一方撮影対象の建物は横幅が 20-30m ほどあるから，この位置変動は 3 次元復元全体で見ると無視できるほど小さい．そこで，RS 歪ありとなしの場合の各系列から SfM を行い，それら復元結果のうちカメラ位置の差を誤差として評価することとした．RS 歪なしの画像に対し普通に SfM を行った場合のカメラ位置を基準に，RS 歪ありの画像に対し，RS 歪モデルありの SfM を行った場合と，モデルなしの通常の SfM を行った場合に，それぞれの差を比較した．Fig.2.10 は 2 つの異なるシーンの結果である．これらの結果より，提案手法で

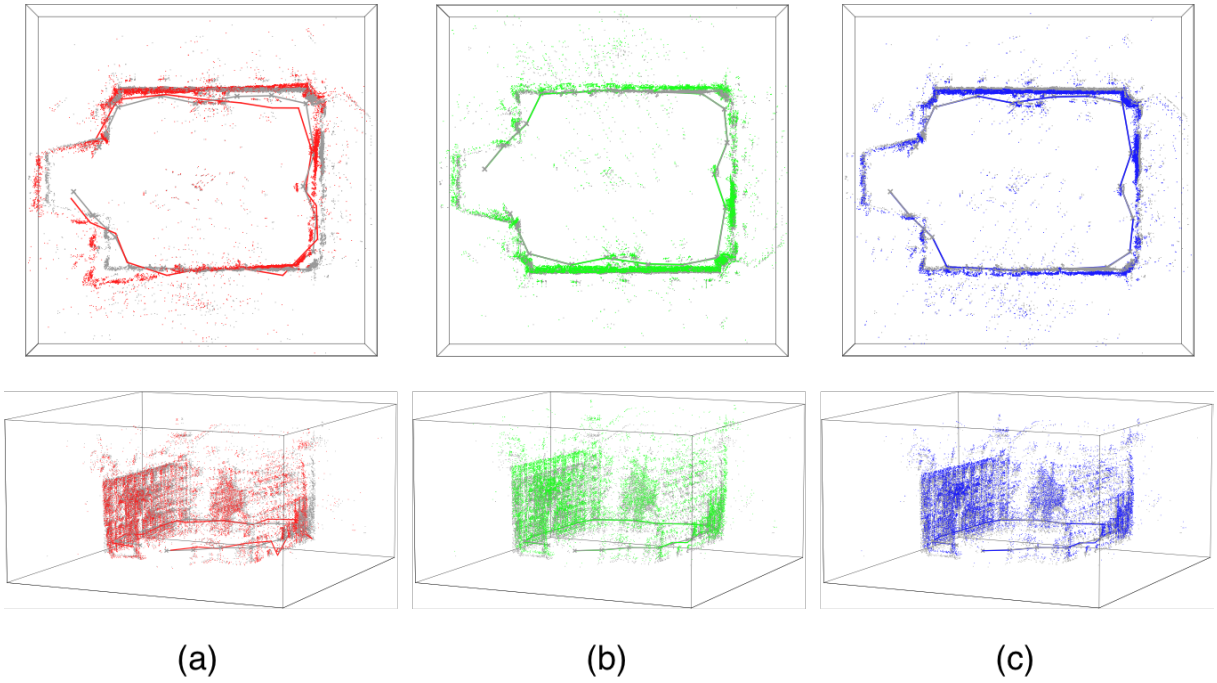


Fig. 2.5: 典型的な復元例：castle-P19. (a) w/o RS. (b) w/ RS. (c) w/ RS* (ϕ_1^1 fixed).

ある近似して簡略化した RS モデルの妥当性について実証できた.

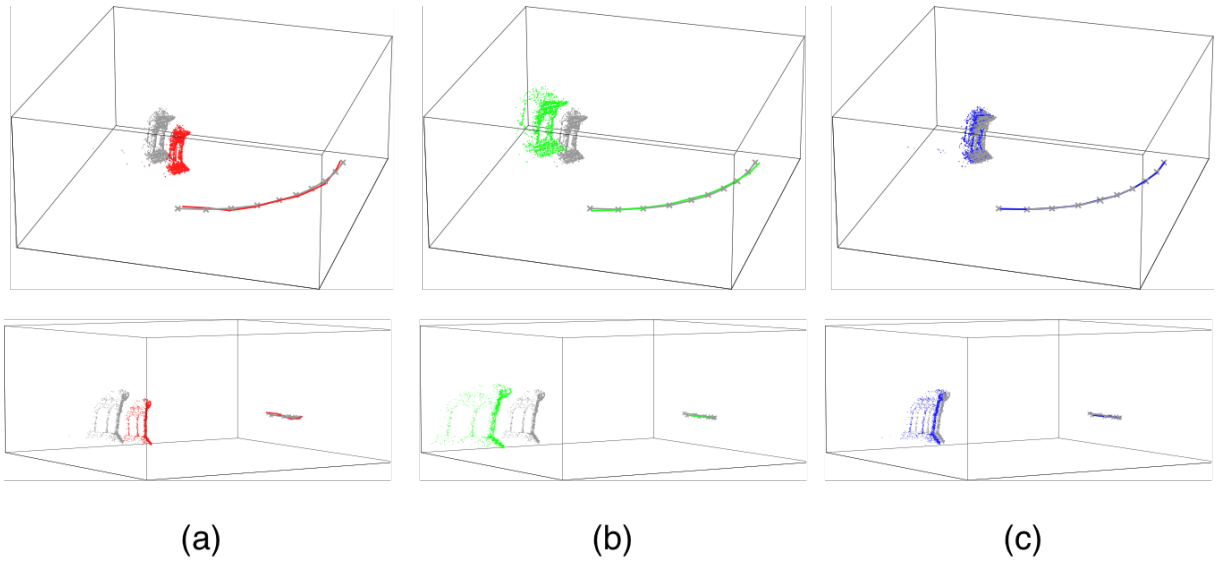


Fig. 2.6: 典型的な復元例：Temple. (a) w/o RS. (b) w/ RS. (c) w/ RS* (ϕ_1^1 fixed).

Fig. 2.7: 実験 II で使われた合成画像の例．上の 3 つがオリジナル画像で下の 3 つが RS 歪みを合成したものになっている．

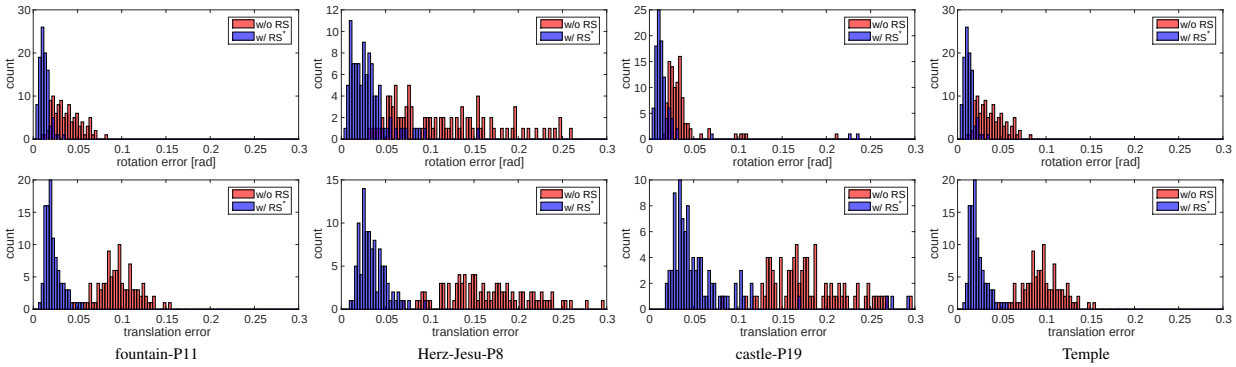


Fig. 2.8: 実験 II の結果．ランダムな RS 歪みを画像に与えて合成して 100 回処理をしたときの誤差ヒストグラム．

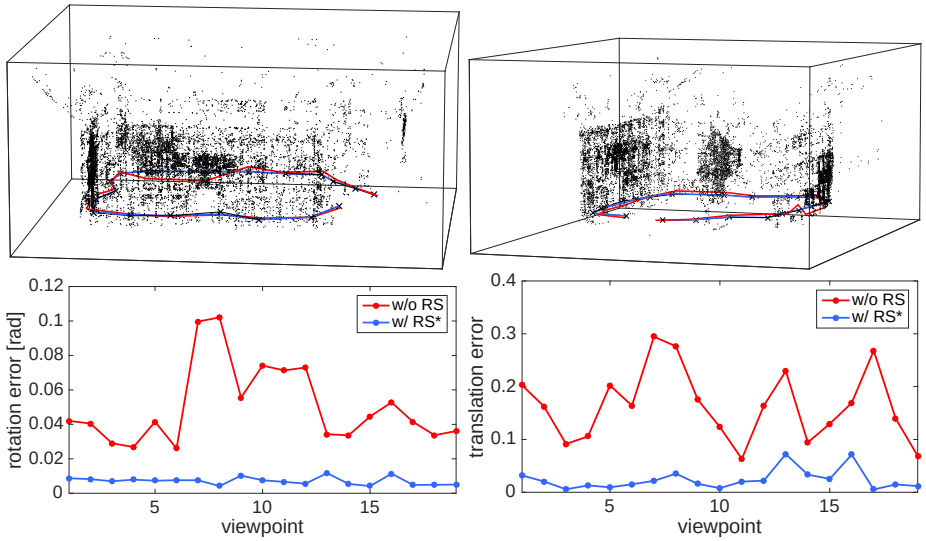


Fig. 2.9: (castle-P19) に RS 歪みを加えた画像から復元された例．歪みなし画像か得られたカメラ位置を真値とみなし，黒の x マークで示した．提案手法による結果は青．通常の BA で得られた結果が赤．推定されたカメラ位置の誤差は歪みなし画像で得られた結果との距離で計算した．

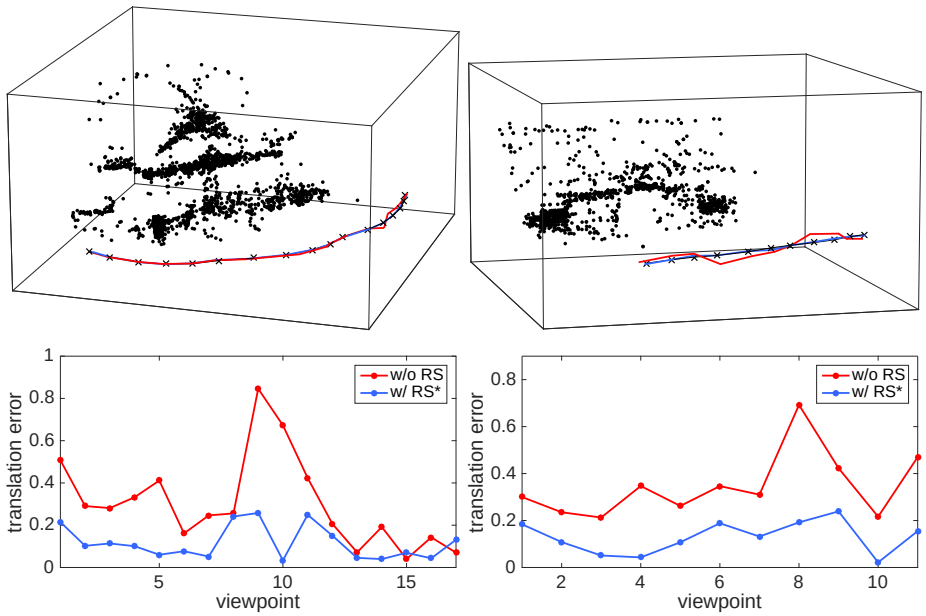


Fig. 2.10: 実画像による 2 つの復元結果．歪みなし画像か得られたカメラ位置を真値とみなし，黒の x マークで示した．提案手法による結果は青．通常の BA で得られた結果が赤．推定されたカメラ位置の誤差は歪みなし画像で得られた結果との距離で計算した．

第3章　平面追跡の高精度化とモデルの拡張

平面追跡には限界があり，対象とする平面がカメラの視線方向に対して完全に垂直を向いたり，平面がカメラから無限遠方に遠ざかる場合，追跡は（当然ながら）不可能である．我々の関心は，追跡が理論的に不可能なこのような条件にどれだけ近いところまで，追跡を行えるかにある．つまり，一定の追跡性能が維持できる限界を，従来方法よりも広げたい．我々の知る限り，同じ動機を持ち，上述の問題を解決しようとした研究はこれまでにない．例えば，平面追跡の手法の評価用にテストデータ [38] が公開されており，そこでは対象平面のテクスチャの多寡や，照明変動やモーションブラーの影響は十分意識されているものの，このような限界性能を試すデータは含まれていない．

本研究のこのような視点は，ほとんどの応用で大事だと思われるが，特に AR やヒューマンインタフェースへの応用で重要である．多様な条件下での様々なユーザによる利用を想定すれば，平面の運動の仕方は，追跡アルゴリズムにとって都合良く制御されたものではあり得ないからである．上述のような限界性能を向上できれば，この種の応用のユーザビリティの向上にも直結すると思われる．

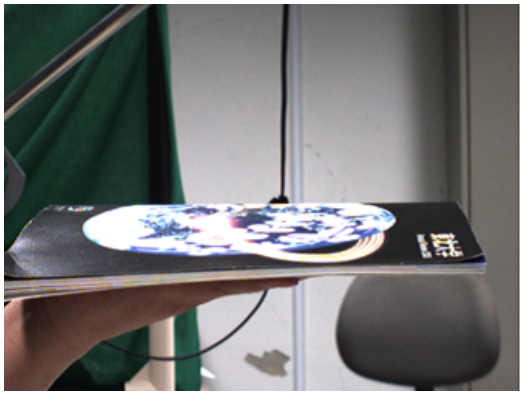


Fig. 3.1: 平面追跡が困難な状況の例．カメラに対して平面が傾いたり，遠くに離れる場合，追跡対象が画像上で小さくなり，平面追跡の性能が劣化する．

この章では，Fig.3.1 のように平面がカメラの視線方向に対して大きく傾いたり，遠方に

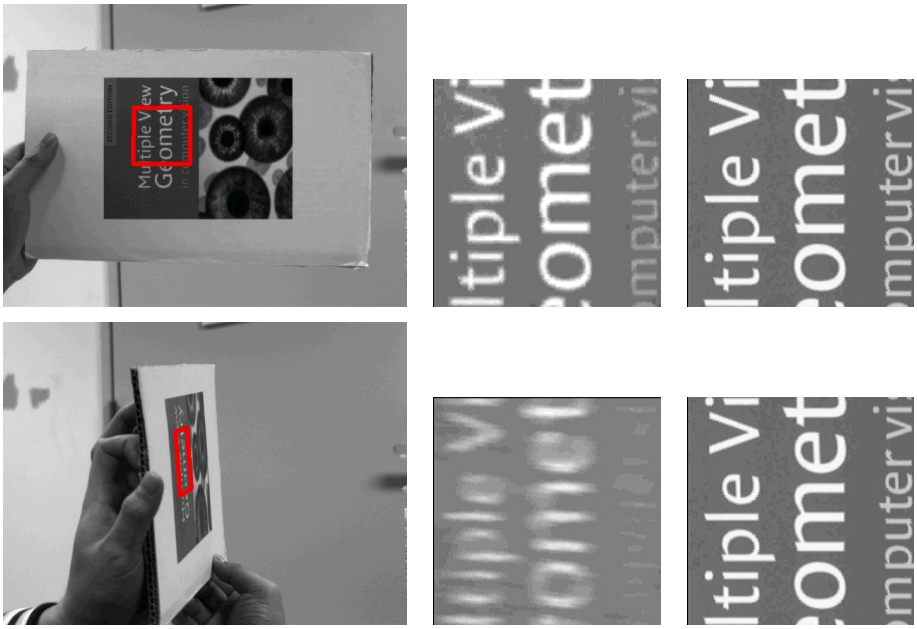


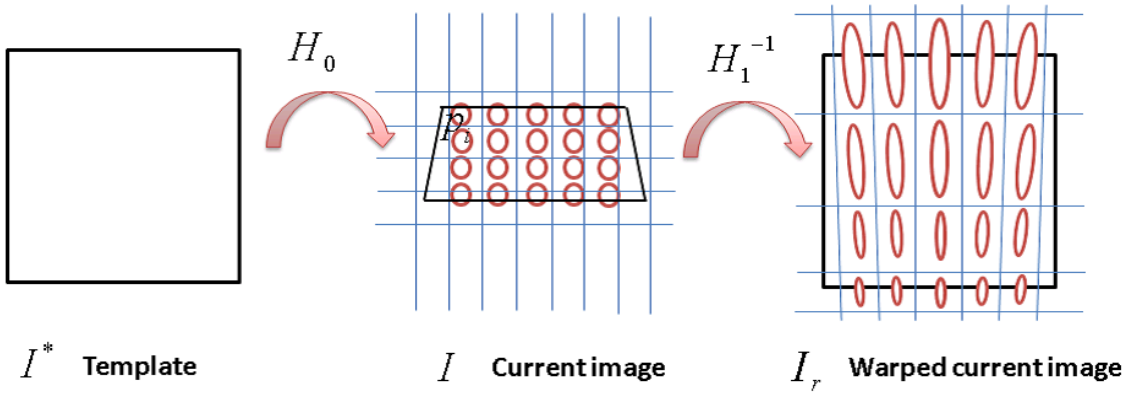
Fig. 3.2: 標本化による画像の実効的解像度低下の例．上段 : 平面がカメラに正対するとき．下段 : 傾いたとき．左列から右列へそれぞれ，追跡中の画像，画像から求めた $\mathcal{I}(w(\mathbf{p}^*;\mathbf{H}))$，およびテンプレート $\mathcal{I}^*(\mathbf{p}^*)$（テンプレートは上下段で同一）．画像の画素数は 640×480，テンプレートは 192×192.

遠ざかる場合に起こる追跡性能劣化の原因を明らかにし，その解決策を示すことを目的とする．加えて，性能を評価するために，新しくベンチマークを作成し，従来手法と提案手法の推定精度・計算速度を比較する．計算モデルの見直しに当たって，計算量の増加が見込まれるため，平面追跡手法を GPU で実装する方法も示し，高速処理を実現する．

3.1 標本化過程を考慮した平面追跡の方法

第 3 章で述べたように，平面追跡では，真の姿勢 $\mathbf{H}_0$ を推定するため，その推定値 $\mathbf{H}_1$ で撮影画像を変形した $\mathcal{I}(\mathbf{H}_1\mathbf{p}^*)$ と，テンプレート $\mathcal{I}^*(\mathbf{p}^*)$ を比較している．Fig.3.2 のように，例えば平面が大きく傾いている場合など，$\mathcal{I}(\mathbf{H}_1\mathbf{p}^*)$ はその変形に応じて，実効的に解像度が低下する．仮に $\mathbf{H}_1=\mathbf{H}_0$ であったとしても，それは $\mathcal{I}^*(\mathbf{p}^*)$ とは一致しなくなる．実際，このような場合には頻繁に追跡性能の低下が見られる．

ここでは，モーションブラーを扱った Jin ら [31] や Mei ら [44] らの方法同様に，このような解像度低下に合わせてテンプレートを適切に修正することで，追跡性能の低下を抑えることを考える．



$$I_r(p^*) = \sum_i \left[I^*(p_i^*) * f(H_0 p_i^*) \right] h(H_1 p^* - H_0 p_i^*)$$

Fig. 3.3: 実効的な解像度低下の生成過程.

3.2 解像度低下のモデル

テンプレートをどのように修正すべきかを考えるために，撮影画像 $\mathcal{I}(\mathbf{p})$ がどのように生成され，さらにテンプレートと比較するためにいかに変形されるかを調べる．

対象とする平面パタン（=テンプレート）上の点 $\mathbf{p}^*$ が写像 $\mathbf{p} \propto \mathbf{H}_0\mathbf{p}^*$ によって画像の点 $\mathbf{p}$ に写るとする．ここで $\mathbf{H}_0$ は真の射影変換である．今，画像撮影時の空間方向の標本化を無視したときの撮影画像を $\mathcal{I}'(\mathbf{p})$ と書くと，これは単に

$$\mathcal{I}'(\mathbf{p}) = \mathcal{I}^*(\mathbf{H}_0^{-1}\mathbf{p}) \tag{3.1}$$

と与えられる．標本化を前提とすると，撮影画像は，プレフィルタ $f(\mathbf{p})$ をこの $\mathcal{I}'(\mathbf{p})$ に適用した連続関数

$$\mathcal{I}''(\mathbf{p}) = \mathcal{I}'(\mathbf{p}) * f(\mathbf{p}) = \mathcal{I}^*(\mathbf{H}_0^{-1}\mathbf{p}) * f(\mathbf{p}) \tag{3.2}$$

を，標本化したものとしてモデル化できる．その標本値は，撮影画像の各画素 $\mathbf{p}_j$ $(j = 1, \ldots)$ での値 $\mathcal{I}''_j \equiv \mathcal{I}''(\mathbf{p}_j)$ である．これを連続領域に再構成したものが撮影画像 $\mathcal{I}(\mathbf{p})$ であるとし，再構成フィルタ $h(\mathbf{p})$ を用いて

$$\mathcal{I}(\mathbf{p}) = \sum_j \mathcal{I}''_j h(\mathbf{p} - \mathbf{p}_j) \tag{3.3}$$

と表す．

Fig.3.3 のように，撮影画像 $\mathcal{I}(\mathbf{p})$ は，テンプレート $\mathcal{I}^*(\mathbf{p}^*)$ と比較すべく変形される．この変形を与える写像を $\mathbf{p} \propto \mathbf{H}_1\mathbf{p}^*$ とする．追跡中，$\mathbf{H}_1$ は $\mathbf{H}_0$ となる．変形後の画像を $\tilde{\mathcal{I}}(\mathbf{p}^*)$

とすると，これは $\tilde{\mathcal{I}}(\mathbf{p}^*) \equiv \mathcal{I}(\mathbf{H}_1\mathbf{p}^*)$ によって与えられ，Eq.(3.3) より

$$\tilde{\mathcal{I}}(\mathbf{p}^*) = \mathcal{I}(\mathbf{H}_1\mathbf{p}^*) = \sum_j \mathcal{I}''_j h(\mathbf{H}_1\mathbf{p}^* - \mathbf{p}_j) \tag{3.4}$$

と与えられる．これに Eq.(3.2) を代入すると

$$\begin{aligned}\tilde{\mathcal{I}}(\mathbf{p}^*) &= \sum_j \left[\mathcal{I}^*(\mathbf{H}_0^{-1}\mathbf{p}_j) * f(\mathbf{p}_j)\right] h(\mathbf{H}_1\mathbf{p}^* - \mathbf{p}_j) \\ &= \sum_j \left[\mathcal{I}^*(\mathbf{p}_j^*) * f(\mathbf{H}_0\mathbf{p}_j^*)\right] h(\mathbf{H}_1\mathbf{p}^* - \mathbf{H}_0\mathbf{p}_j^*)\end{aligned} \tag{3.5}$$

を得る．ただし 2 番目の等式は $\mathbf{p}_j^*$ を $\mathbf{p}_j^* \equiv \mathbf{H}_0\mathbf{p}_j$ と定義し，$\mathbf{p}_j$ を置換して得られたものである．

3.3 線形フィルタの畳込みによる近似

Eq.(3.5) は，$\mathbf{H}_0$ の姿勢をとる平面を撮影した画像を，$\mathbf{H}_1$ で変形して得られるもののモデルである．従来手法のテンプレートをこれで置き換えれば目的は果たされる．Eq.(3.5) 中の $\mathbf{H}_0$ は不明なので，まずこれを $\mathbf{H}_1$ で近似し，その後 $\mathbf{H}_1$ を $\hat{\mathbf{H}}\mathbf{H}(\mathbf{x})$ で置き換えたものを $\tilde{\mathcal{I}}_{\hat{\mathbf{H}}\mathbf{H}(\mathbf{x})}(\mathbf{p}^*)$ と記すことにすれば，更新量 $\mathbf{x}$ を

$$\mathbf{x}_o = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \left[\mathcal{I}(\hat{\mathbf{H}}\mathbf{H}(\mathbf{x})\mathbf{p}_i^*) - \tilde{\mathcal{I}}_{\hat{\mathbf{H}}\mathbf{H}(\mathbf{x})}(\mathbf{p}_i^*)\right]^2 \tag{3.6}$$

によって定めればよい．

しかしながら実時間性を前提とすると，Eq.(3.6) の直接計算はあまり現実的でない．$\tilde{\mathcal{I}}(\mathbf{p}^*)$ は撮影画像の標本グリッド $\mathbf{p}_j^*$ についての和を計算するし，また未知数を含むフィルタを適用した後での標本化を要するからである．そこで，Eq.(3.5) の $\tilde{\mathcal{I}}(\mathbf{p}^*)$ を，テンプレート $\mathcal{I}^*(\mathbf{p}^*)$ に線形フィルタ $g(\mathbf{p}^*; \hat{\mathbf{H}}\mathbf{H}(\mathbf{x}))$ を畳込んだ

$$\tilde{\mathcal{I}}(\mathbf{p}^*) \approx \mathcal{I}^*(\mathbf{p}^*) * g(\mathbf{p}^*; \hat{\mathbf{H}}\mathbf{H}(\mathbf{x})) \tag{3.7}$$

のように，近似的に表すことを考える．もし十分な精度でこれが可能であれば，従来手法に対し計算量の増加を最小限に抑えられる．$g(\mathbf{p}^*; \hat{\mathbf{H}}\mathbf{H}(\mathbf{x}))$ を一様なフィルタとして表すには，$\hat{\mathbf{H}}\mathbf{h}(\mathbf{x})$ を Fig.3.4 のようにアフィン変換で近似する必要がある．計算の詳細は後程示すことにする．

この近似は非一様な解像度低下を表現することはできないが，非等方性の解像度低下は表現することが可能であり，これによって幾何学量の推定をするのに十分な精度が得られると我々は考えている．

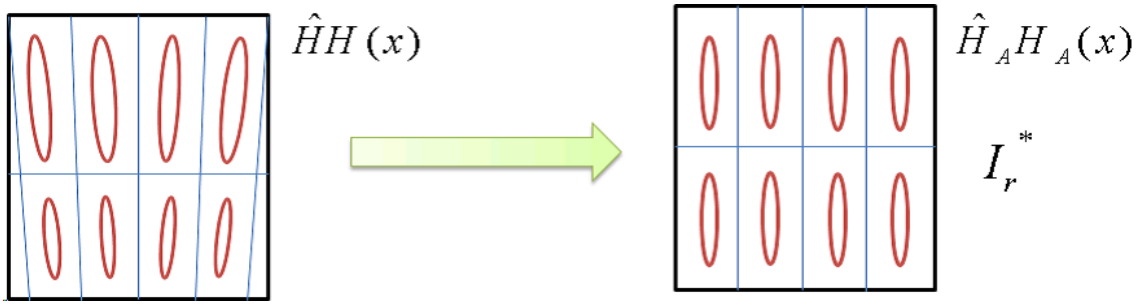


Fig. 3.4: 非一様なフィルタを線形フィルタとして近似.

これからフィルタ g を求めていく．テンプレート $\mathcal{I}^*$ が離散データであり，各画素 $\mathbf{p}_i^*(i = 1,\ldots)$ での標本値 $\mathcal{I}_i^*$ が与えられているとすると，連続領域の $\mathcal{I}^*(\mathbf{p}^*)$ は，再構成フィルタ $h(\mathbf{p}^*)$ を用いて

$$\mathcal{I}^*(\mathbf{p}^*) = \sum_i \mathcal{I}_i^* h(\mathbf{p}^* - \mathbf{p}_i^*) \tag{3.8}$$

と表せる．この式と Eq.(3.5) を Eq.(3.6) に当てはめて，$\mathbf{p}_i^*$ を $\mathbf{p}_j^*$ と近似的に同一視すると

$$h(\mathbf{p}^* - \mathbf{p}_i^*) * g(\mathbf{p}^*; \tilde{\mathbf{H}}) = h(\tilde{\mathbf{H}}(\mathbf{p}^* - \mathbf{p}_i^*)) \tag{3.9}$$

の関係を得る．ただし，$\tilde{\mathbf{H}} \equiv \hat{\mathbf{H}}\mathbf{H}(\mathbf{x})$ であり，また上ではプレフィルタ f の効果を無視した．この式を満たす g を次のように計算する．再構成フィルタをガウス関数

$$h(\mathbf{p}^*) \propto \exp\left(-\frac{1}{2\sigma^2}\mathbf{p}^{*\top}\mathbf{p}^*\right) \tag{3.10}$$

とする．ここで exp 関数内の $\mathbf{p}^*$ は，非同次座標のベクトル（i.e., $\mathbf{p}^* = [x^*, y^*]^\top$）である．次に，$\tilde{\mathbf{H}}$ のアフィン近似し，左上 2×2 の部分行列を $\tilde{\mathbf{H}}_A$ とする．そのとき Eq.(3.9) の右辺 $h(\tilde{\mathbf{H}}(\mathbf{p}^* - \mathbf{p}_i^*))$ は，

$$h(\tilde{\mathbf{H}}(\mathbf{p}^* - \mathbf{p}_i^*)) \propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{p}^* - \mathbf{p}_i^*)^\top \tilde{\mathbf{H}}_A^\top \tilde{\mathbf{H}}_A (\mathbf{p}^* - \mathbf{p}_i^*)\right). \tag{3.11}$$

と書ける．Eq.(3.9) の関係とガウス関数の畳込みの性質（$N(\mathbf{a},\mathbf{A}) * N(\mathbf{b},\mathbf{B}) = N(\mathbf{a}+\mathbf{b}, \mathbf{A}+\mathbf{B})$）から，

$$g(\mathbf{p}^*; \tilde{\mathbf{H}}) \propto \exp\left(-\frac{1}{2\sigma^2}\mathbf{p}^{*\top}(\tilde{\mathbf{H}}_A^{-1}\tilde{\mathbf{H}}_A^{-\top} - \mathbf{I})^{-1}\mathbf{p}^*\right). \tag{3.12}$$

を得る．

以上をまとめると，更新量の計算式は

$$\mathbf{x}_o = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \left[\mathcal{I}(\hat{\mathbf{H}}\mathbf{H}(\mathbf{x})\mathbf{p}_i^*) - \mathcal{I}^*(\mathbf{p}_i^*) * g(\mathbf{p}_i^*; \hat{\mathbf{H}}\mathbf{H}(\mathbf{x}))\right]^2 \tag{3.13}$$

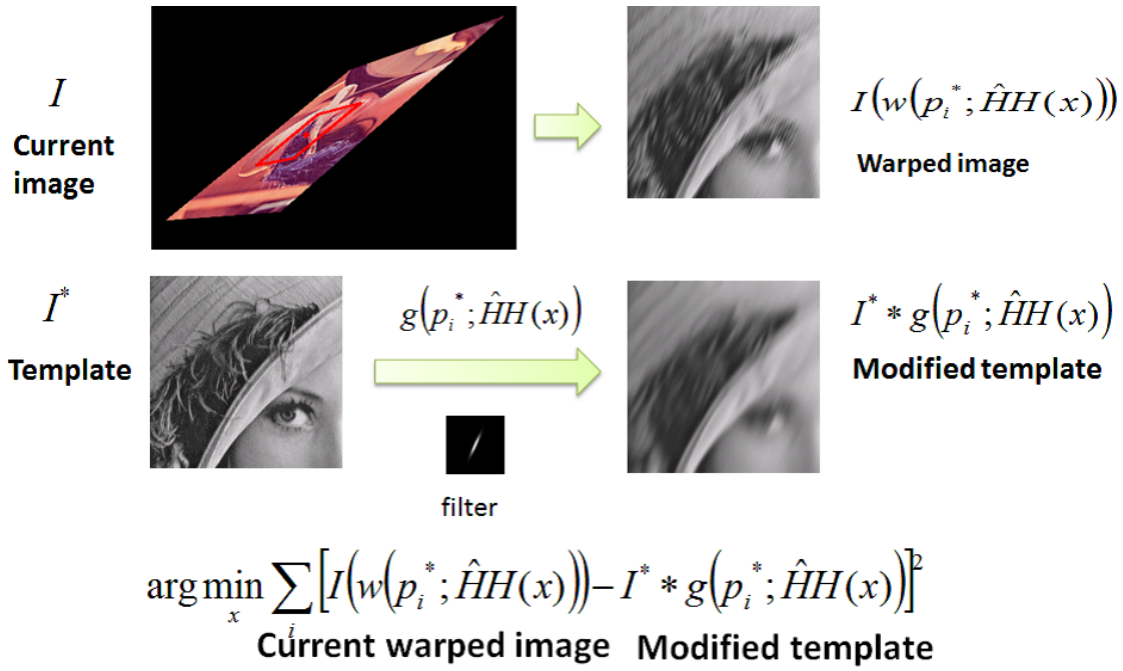


Fig. 3.5: 標本化過程を考慮した最適化の概略図．ここではシミュレーション画像でフィルタの推定とテンプレートの更新を行っている．

となる．この式は，前述したモーションブラーの場合の Eq.(1.69) 同様，差分をとる 2 項がともに変数 **x** に依存しており，収束性に懸念がある．今の場合，テンプレートの修正は，現在の平面姿勢の瞬時値にしか依存しない．しかも，通常，反復一回の修正量である **x** は微小であるから，その **H**(**x**) によるテンプレートの修正量はわずかである．そこで，テンプレートの修正量を近似し，次のように **x** に依存しないようにする．

$$\mathbf{x}_o = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \left[I(\hat{\mathbf{H}}\mathbf{H}(\mathbf{x})\mathbf{p}_i^*) - I^*(\mathbf{p}_i^*) * g(\mathbf{p}_i^*; \hat{\mathbf{H}}) \right]^2 \tag{3.14}$$

この近似は平面の動きが極端に速くない限り十分に成立する．なおモーションブラーの場合，テンプレートの修正は各フレーム間隔内での平面の微小運動の積分値に依存するから，このような近似は不可能である．Fig.3.5 に最適化の概略図を乗せた．ここでは，シミュレーション画像をつかって平面トラッキングを行っている．

モーションブラーの場合や提案手法は，テンプレートを更新が必要なため，Inverse Compositonal 法にはあまり向いていない．なぜなら，この方法はテンプレートを固定して，ニュートン法の反復計算で必要なヘッセ行列を事前に計算し，計算量を抑えているからである．

3.4 実装の詳細

これまでの流れを要約すると，我々は Eq.(3.14) に従い，反復計算によって $\mathbf{x}_o$ を推定している．フィルタ g は各フレームで更新され，フレーム内の反復処理では固定されたままになっている．フィルタ g を得るため，はじめに $\hat{\mathbf{H}}$ のアフィン近似を行う（$\hat{\mathbf{H}}$ によって射影される点とアフィン変換によって射影される点の二乗和を最小にするパラメータを使用した）．そして，$\hat{\mathbf{H}}_A$ をアフィン変換の左上 2×2 行列とするとし，g を

$$g(\mathbf{p}^*;\hat{\mathbf{H}}) \propto \exp\left(-\frac{1}{2\sigma^2}\mathbf{p}^{*\top}(\hat{\mathbf{H}}_A^{-1}\hat{\mathbf{H}}_A^{-\top}-\mathbf{I})^{-1}\mathbf{p}^*\right), \tag{3.15}$$

とした．ここで，$\mathbf{p}^*$ は非同次ベクトルである．

これまで，我々は画像の解像度が単純に低下する場合だけを考えてきたが，もし解像度が変化しないならば，上式の行列 $\hat{\mathbf{H}}_A^{-1}\hat{\mathbf{H}}_A^{-\top}-\mathbf{I}$ は $\mathbf{O}$ になる．このような状況で，g は理想的にはデルタ関数を取る．また，平面がカメラに対して近づき解像度が増加した場合，この行列はもはや正定値ではなくなり，意味のある解を持たない．したがって我々は，$\hat{\mathbf{H}}_A^{-1}\hat{\mathbf{H}}_A^{-\top}-\mathbf{I}$ が正定値になるよう制約をかけ，解像度が低下する場合にのみテンプレートを更新するようにした．まず，$\hat{\mathbf{H}}_A^{-1}\hat{\mathbf{H}}_A^{-\top}-\mathbf{I}=\lambda_1\mathbf{e}_1\mathbf{e}_1^\top+\lambda_2\mathbf{e}_2\mathbf{e}_2^\top$ と固有値分解し

$$g(\mathbf{p}^*;\hat{\mathbf{H}}) \propto \exp\left(-\frac{1}{2\sigma^2}\mathbf{p}^{*\top}\left(\max(\lambda_1,\varepsilon)\mathbf{e}_1\mathbf{e}_1^\top+\max(\lambda_2,\varepsilon)\mathbf{e}_2\mathbf{e}_2^\top\right)^{-1}\mathbf{p}^*\right). \tag{3.16}$$

のように計算した．ここで，我々は $\varepsilon=0.01$ とした．

これから，案手法の計算量の評価について述べる．我々の実装では，拡張した ESM 法（光の変化への対処 [57] とモーションブラーへの対処 [49] を施した）に提案手法を加えて計算している．提案手法では，カメラのフレームレートや平面の動きの大きさに依存して，大体 3〜20 回の反復で解が収束している．テンプレートの更新は，フィルタ g の計算とテンプレートへの畳み込みによって構成されており，フレームの初めに処理が行われる．フィルタのサイズが 49×49 と比較的大きいため，テンプレートの畳み込みは，FFT によって周波数領域に変換してから計算している．我々は計算の大部分を GPU 上で行うことで，テンプレートサイズ 192×192，フィルタサイズ 49×49 のコンボリューションを約 1.2ms で計算している．この条件では，一回の反復計算に 0.4ms かかるため，計算量の増加は 1 フレームあたり 3 反復分となる．

3.5 実験方法

提案手法の有効性を調べるため，いくつかの実験を行った．実験では，前節で述べた提案手法と従来手法を比較した．従来手法は，Malis らの ESM 法に照明変化モデル [57]

および Park らのモーションブラーのモデル [49] を加えた方法と，MI を類似度の基準として追跡を行う方法 [13] である．提案手法は，従来で用いた ESM に解像度低下のモデル Eq.(3.14) をさらに加えたものである．

3.5.1 実験装置

追跡対象の平面が視線方向に対して大きく傾いた場合の

- 追跡の安定性・頑健性および
- 変形パラメータの推定精度

が実用上重要である．これらを評価するため，3 枚の平面が互いに正確に 90 度の角度（角度誤差は 0.01 度以上）をなすように機械加工した物体を用いた（Fig.3.6 に全体像がある）．垂直をなす 2 枚の平面を選び，片方には追跡対象となる平面パタンを，もう一方にはチェスボードパタン（11×8）を貼り付けた．この平面パタンを提案手法（および比較対象とする従来手法）にて追跡し，その推定精度をチェスボードパタンを使って求めたものと比較した．平面パタンとチェスボードパタンが互いに直交することから，平面パタンが視線に対して大きく傾くとき，チェスボードパタンは逆に視線に正対し，その平面姿勢推定の精度はほぼ最良となるのでグランドトゥルースとするにふさわしい．

その他の装置は次の通りである．毎秒 60 フレームでサイズ 640×480 の画像を撮影する Point Grey Research 社の Grasshopper をカメラに， 3.2GHz の Core i7 を CPU に持ち，nVidia 社の GTX480 を GPU に持つ PC を用いた．主要な計算は GPU 上で行うことで，カメラの画像転送にかかる時間を除いた正味の計算時間は 3 から 4 ミリ秒程度であり，実時間性は十分ある．

3.5.2 精度の評価方法

精度の評価は，平面追跡の結果得られる 2 次元射影変換 $\mathbf{H}$ そのものではなく，それを元に計算される平面パタンの空間姿勢を用いて行った．これは，平面の姿勢の方が誤差をより直観的に理解でき，またそれが主に AR を中心とする応用で最終的に必要とされるものであるからである．

平面の姿勢を求めるには，平面パタン，チェスボードパタンいずれの場合も，カメラの内部パラメータをあらかじめ校正しておく必要があり，ここでは Zhang の方法 [69] を利用した．平面パタンの場合，平面パタン上に定義した 3 次元座標系の点 $\mathbf{x}$ から画像座標系

Fig. 3.6: 精度の評価に用いた実験装置．

の点 (u, v) への投影を $[u, v, 1]^\top \propto \mathbf{K}(\mathbf{R}\mathbf{x} + \mathbf{t})$ と書くと，$\mathbf{H} \propto \mathbf{K}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}]$ を得る．ここで $\mathbf{K}$ は，校正したカメラの内部パラメータからなる 3×3 行列で，$\mathbf{r}_1$ および $\mathbf{r}_2$ は，回転行列 $\mathbf{R}$ の 1，2 列ベクトルである．この関係から，$\mathbf{K}^{-1}\mathbf{H}$ を計算し，その 3×3 行列の第 3 列ベクトルを $\mathbf{t}$ の推定値とし，第 1, 2 列ベクトル $\mathbf{q}_1$，$\mathbf{q}_2$ を用いて新たに 3×3 行列 $[\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_1\times\mathbf{q}_2]$ を作成し，その特異値分解における特異値をすべて 1 に置き換えて計算される行列を，$\mathbf{R}$ の推定値とした．この方法は内部パラメータが既知であることを前提とすれば必ずしも最適なものとは言えない[1]ものの，平面追跡のアルゴリズムは一般に $\mathbf{H}$ を推定するように定式化されていることから，このようにした，また，少なくとも 2 つの方法の性能を公平に比較する目的では問題ないはずである．

一方，チェスボードパタンから平面の姿勢を推定する方法は，チェスボードのコーナー点とその画像上の位置の対応を使って，再投影誤差が最小になるようにチェスボードの位置姿勢（上述の $\mathbf{R}$ と $\mathbf{t}$ にあたる）を推定し，平面パタンがチェスボードと直交する事実を用いて，これを平面パタンの位置姿勢に換算した．

[1] 平面追跡の定式化の際，$\mathbf{H}$ の代わりに $\mathbf{R}$，$\mathbf{t}$ を未知パラメータとする方がより高精度だろう．

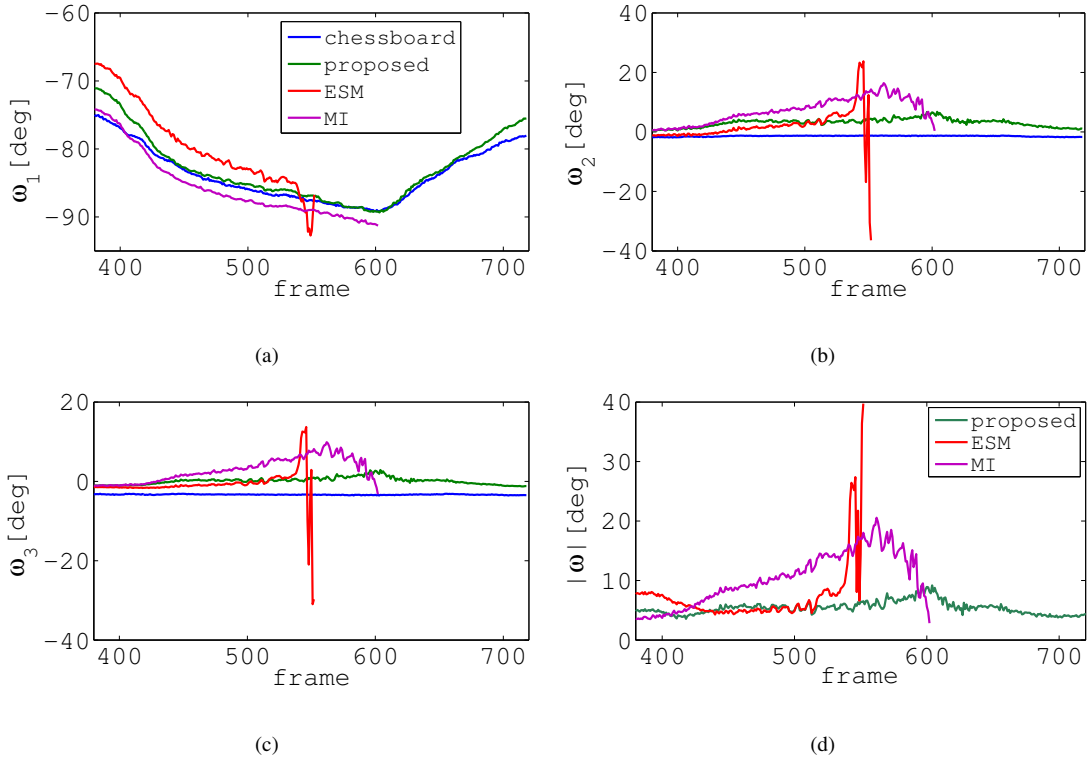


Fig. 3.7: 各方法（青 : チェスボード，緑 : 提案手法，赤 : ESM，紫 : MI）で推定した平面姿勢の回転成分の時間変化．(a)〜(c) は回転を回転軸・角度表現したベクトルの 3 成分． (d) はチェスボードによる推定とのずれを角度の大きさで表したもの．従来手法である ESM と MI は途中のフレームで追跡が失敗している．

3.6 実験結果

3.6.1 精度評価実験

Fig.3.8 に示すような平面パタンを対象に追跡実験を行った．Fig.3.7 にその追跡結果を示す．Fig.3.8 は，Fig.3.7 中の 3 つの時点（第 400, 527, 547 フレーム目）におけるスナップショットである．テンプレートサイズは 160×160 画素である．図は，上述のように計算した平面の姿勢 $\mathbf{R}$ および $\mathbf{t}$ のうち，特に提案手法と従来手法との間で差が顕著に現れる $\mathbf{R}$ の時間変化を示す．図の左の 3 つのプロットは，各方法で推定した $\mathbf{R}$ を回転軸・角度表現した 3 次元ベクトル $[\omega_1, \omega_2, \omega_3]$ の各成分にあたる．一番右のプロットは，チェスボードパタンで推定したものとの $\mathbf{R}$ のずれを，回転角の大きさに直して表示したものである．なお，すべてのプロットの単位は度（degree）である．この画像系列では，平面を，視線に対する角度が 70 度付近から 90 度付近まで動かし，戻している（Fig.3.7 の ω_1 がほぼこの角度に相当する）．

ESM を使った方法は，550 フレーム目手前で大きな誤差を生じた後，追跡に失敗した．MI を使った方法は，さらに 50 フレーム追跡し，その後失敗した．一方，提案手法は全系列で追跡に成功している．そこに至る過程で推定精度をチェスボードのそれと比較すると，平面の傾きが小さい場合，従来手法と提案手法は似たような精度を示すが，傾きが大きくなったときの振る舞いが大きく異なり，提案手法が明確に勝る．これは Fig.3.8 を見ても確かめられる．同図 (a)-(d) には，推定された平面姿勢を使って平面上にとった座標フレームを視覚化したものを表示してある．同図 (a), (b)（提案手法）と (c), (d)（従来手法）を比べると，上段（400 フレーム），中段（527 フレーム）では両者にほとんど違いはないが，下段（547 フレーム）の結果において，従来手法による推定に大きな誤差が含まれることが見て取れる．また Fig.3.8 には，画像を推定パラメータで変換したものを (f) に，解像度低下のモデルに基づいて修正したテンプレートを (g) にそれぞれ示している．両者は各フレームである程度近く，提案した解像度低下のモデルの妥当性を裏付ける．(h) はテンプレート修正に用いた線形フィルタ g であるが，平面追跡の結果に応じて変化していることが分かる．

別な平面パタンに対して撮影した画像系列の結果を Fig.3.9 に示す．最初の平面パタン（テンプレート）は比較的滑らかな濃淡変化を示したのに対し，こちらのそれ（Fig.3.10(e)）はシャープなエッジのみで構成されている点で違いがある．Fig.3.10 に特定の 3 フレームでの画像および推定姿勢を可視化したものを示す．結果は若干の違いは認められるものの，Fig.3.7 の系列とほぼ同様である．ESM を使った方法は 400 フレーム手前で追跡に失敗し，その直前での精度が大きく低下していることが認められる．MI を使った方法は，ESM よりもさらに前のフレームで追跡が失敗している．一方，提案手法は全フレームで追跡を行えている．ただし Fig.3.7 の系列とは異なり，平面の視線に対する傾きが 90 度に近い領域で，姿勢の推定精度のばらつきが大きくなっている．これは平面パタンの濃淡構造によるものと考えられる．しかし，推定のバイアスは大きくなっていないこと，追跡自体は継続できているため，必要ならば時間方向に平滑化することで実行的な精度を向上させることも可能だろう．

実世界の平面に仮想物体を重ね込む AR では，追跡対象の平面が視線に対し大きく傾いたり遠くに配置される状況が想定される．提案手法はそのような場合に最も効果を発揮する．例を Fig.3.11 に示す．従来手法で精度良く追跡できないような場合でも，提案手法は安定して高精度に追跡でき，AR の表示品質を高く保つことができると分かる．

3.7 追跡安定性

平面を手で自由に動かして約 1000 フレームからなる画像系列を得，従来手法と提案手法をこれに適用した．結果を Fig.3.12 に示す．従来手法の ESM はこのうち 4 フレームで追跡に失敗した．MI を使った方法では，図中の最初のフレームで追跡できたが，残りの 3 フレームで失敗し提案手法は全フレームで安定して追跡を行えた．

このとき，収束にかかった反復回数の比較結果を Fig.3.13 に示す．解像度低下の影響をモデル化していない ESM では，平面が大きく傾いたときに反復回数が増大し，収束していないことがわかるが，提案手法は，追跡が困難な領域に対しても，計算が収束する．一回のテンプレートの更新にかかる時間は反復計算 3 回分なので，解像度の影響を大きく受けるような状況において，提案手法のほうが処理時間を短縮できる．

Fig.3.12 と同様の実験を，画像系列を変えて行った時の比較結果を Fig.3.14 に，反復回数を Fig.3.15 に示す．ESM と MI はどちらも，Fig.3.14 の 3 箇所で追跡に失敗していた．一方，提案手法はすべてのフレームで追跡可能であった．

収束にかかった反復回数 Fig.3.15 を比較すると，提案手法の方がはるかに少ない反復回数で解が収束していた．これは，シャープなエッジで画像が構成されている場合，特に解像度低下の及ぼす影響が大きいからだと考えられる．

また，Fig.3.17 に別の画像系列に対する結果を示す．この系列では，対象平面はカメラから徐々に遠ざかる．従来手法は，ある距離以上のところで追跡に失敗した．一方で提案手法は，それよりはるかに遠い距離でも正しく追跡できた．なお，この系列では，追跡対象の平面パタンに [38] に収録されている画像を用いた．

3.8 AR への応用例

実世界の平面に仮想物体を重ね込む AR では，追跡対象の平面が視線に対し大きく傾いたり遠くに配置される状況が想定される．提案手法はそのような場合に最も効果を発揮する．その例として Fig.3.11 および Fig.3.17 を示す．従来手法で精度良く追跡できないような場合でも，提案手法は安定して高精度に追跡が可能である．

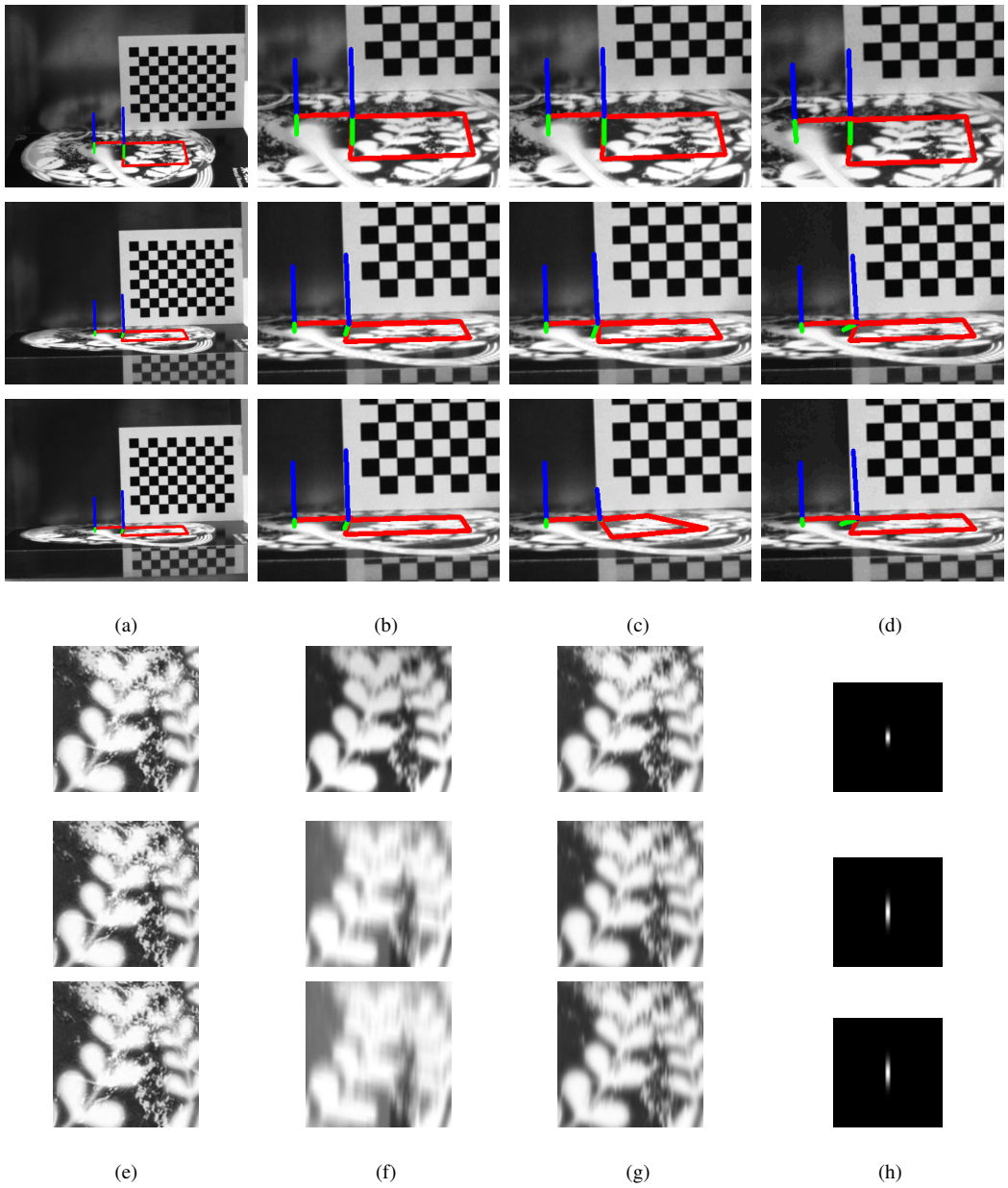


Fig. 3.8: Fig.3.7 の画像系列の追跡結果のスナップショット． 上段，中段，下段はそれぞれ第 400, 527, 547 の 3 フレーム． 各列 (a) から (h) は次の通り．(a) 提案手法の追跡結果（赤い四角形）と，そこから復元した平面上の座標フレーム（赤青緑 3 色の線分で座標軸を表示，右側の大きな方），およびチェスボードパタンから推定した平面上の座標フレーム（同様に 3 色で座標軸を表示，左側の小さな方．ただし見やすいように平面上で並行移動した）．(b) これを拡大表示したもの．(c) ESM の結果．(d) MI の結果．(e) 追跡したテンプレート (160 × 160 pixels; 上中下段で同一)． (f) 収束後の $\mathcal{I}(\hat{\mathbf{H}}\mathbf{p}^*)$．(g) 提案手法による修正後のテンプレート． (h) テンプレートの修正に用いたフィルタ (49 × 49 pixels)．

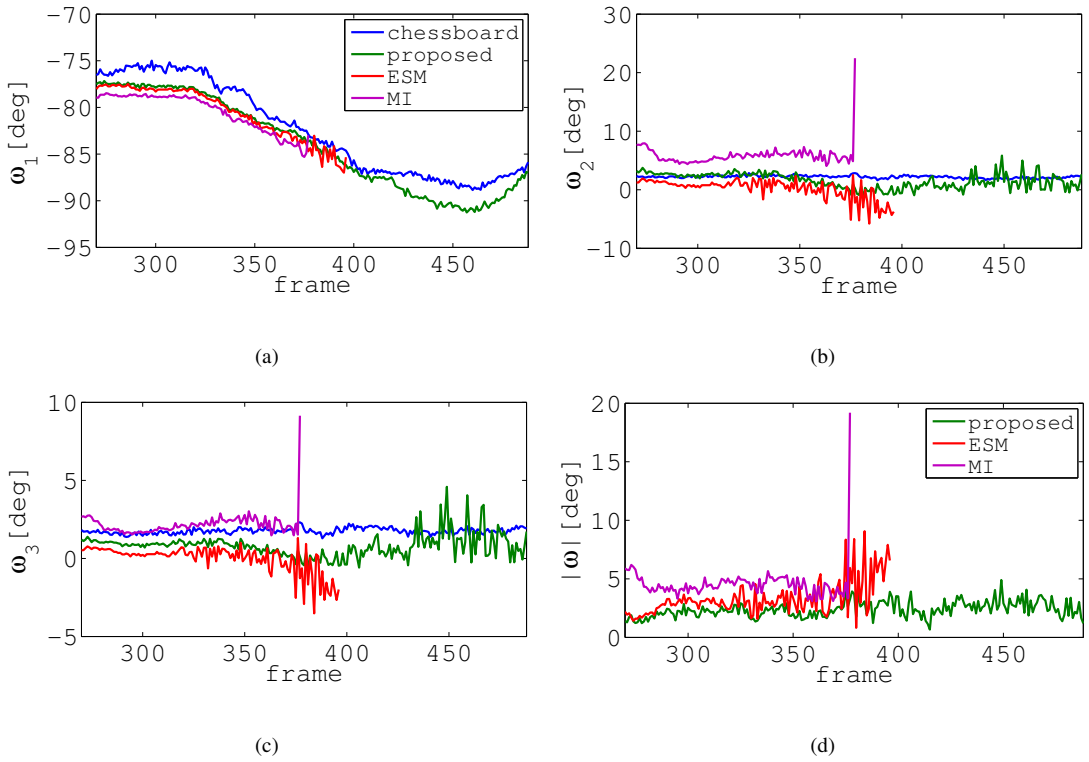


Fig. 3.9: 別の画像系列に対し，各方法で推定した平面姿勢の回転成分の時間変化．図の配置は Fig.3.7 と同じ．

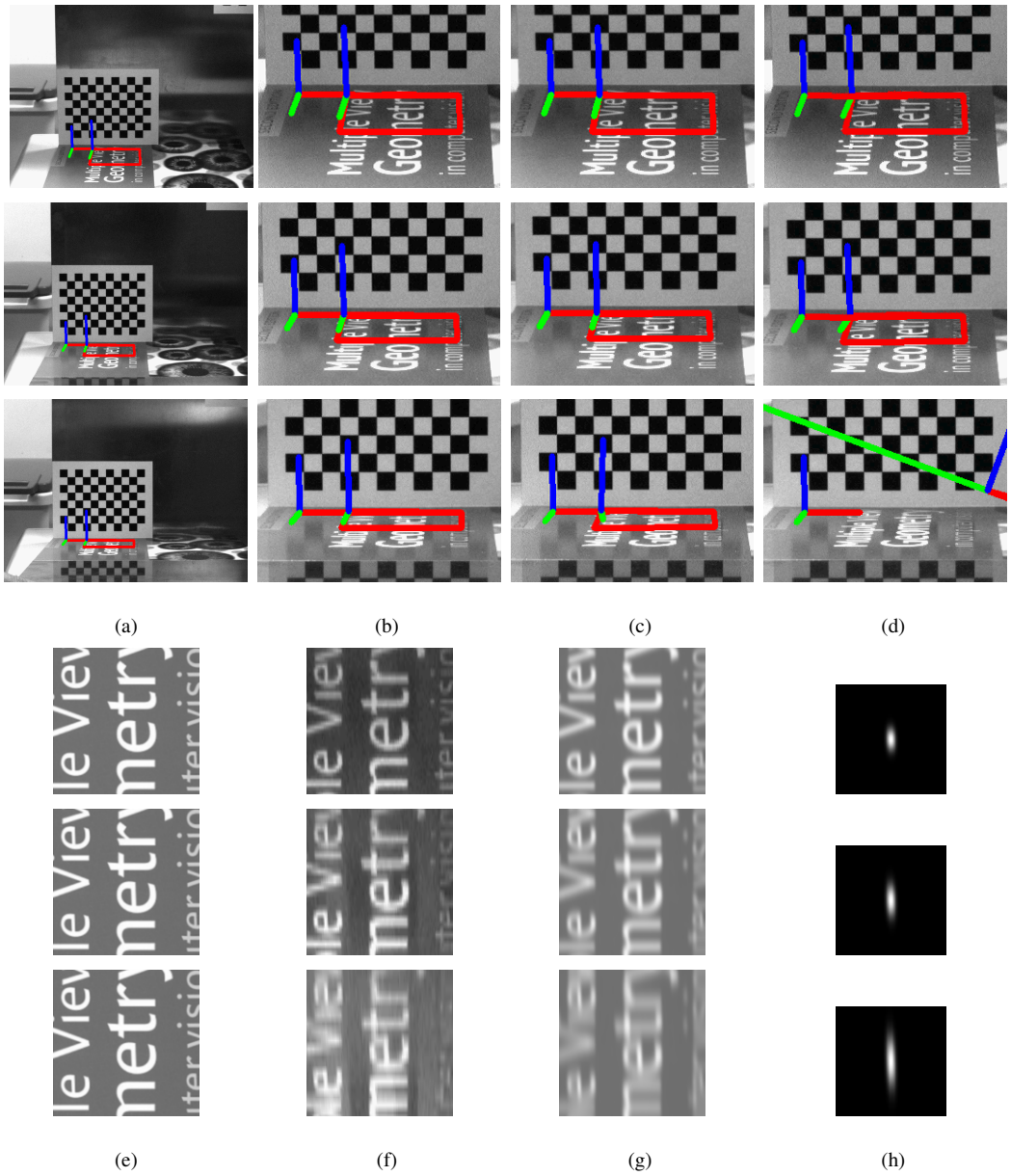


Fig. 3.10: Fig.3.9 の画像系列の追跡結果のスナップショット． 上段，中段，下段はそれぞれ第 297, 360, 395 の 3 フレーム． 各列 (a) から (h) は次の通り．(a) 提案手法の追跡結果（赤い四角形）と，そこから復元した平面上の座標フレーム（赤青緑 3 色の線分で座標軸を表示，右側の大きな方），およびチェスボードパタンから推定した平面上の座標フレーム（同様に 3 色で座標軸を表示，左側の小さな方．ただし見やすいように平面上で並行移動した）．(b) これを拡大表示したもの．(c) ESM の結果．(d) MI の結果．(e) 追跡したテンプレート． (f) 収束後の $\mathcal{I}(\hat{\mathbf{H}}\mathbf{p}^*)$．(g) 提案手法による修正後のテンプレート． (h) テンプレートの修正に用いたフィルタ ．

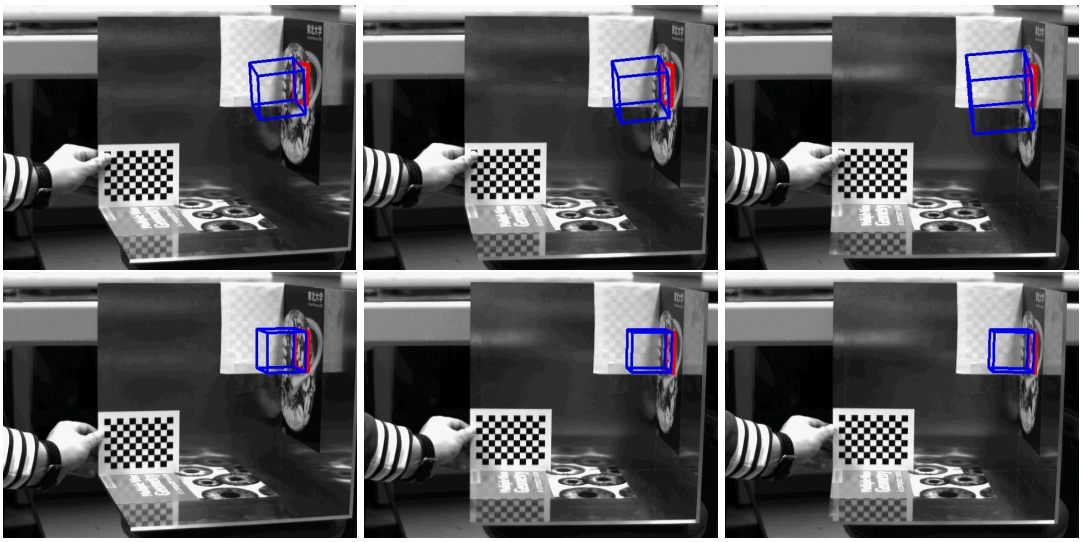


Fig. 3.11: 平面の追跡結果に基づいて立方体を重畳したもの．上段：従来手法．下段：提案手法.

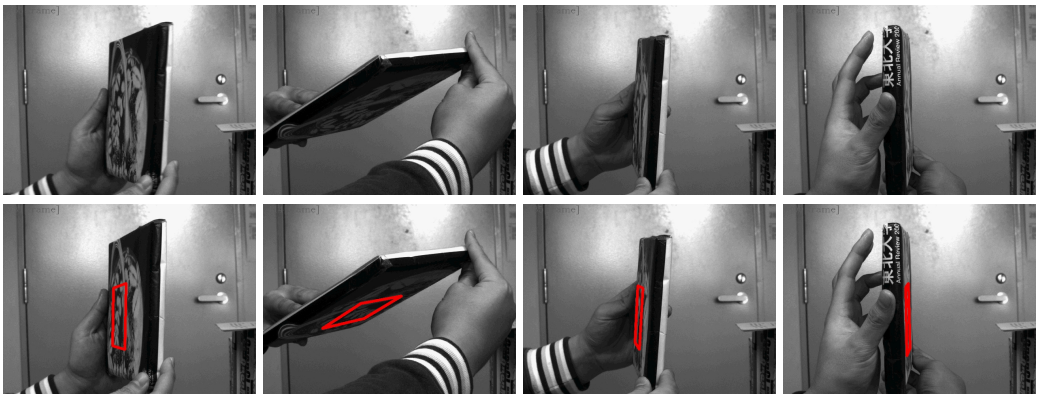


Fig. 3.12: ある画像系列に対して，従来手法で追跡を継続できなかった画像（上段）．提案手法は系列を通して追跡を行えた（下段）.

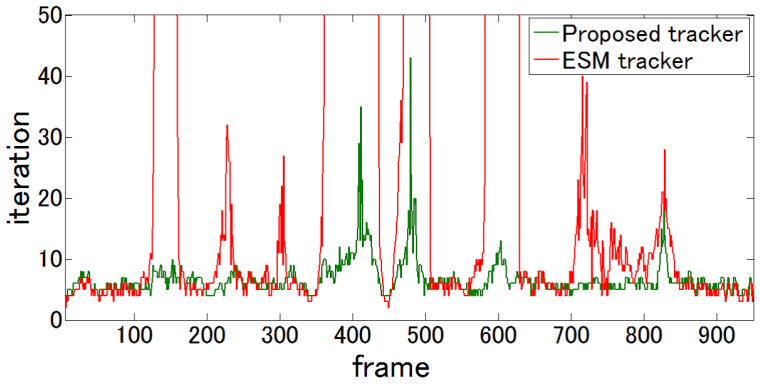


Fig. 3.13: Fig.3.12 の実験における，収束にかかった反復回数の比較．赤が解像度低下の影響を考慮していない ESM，緑が提案手法．

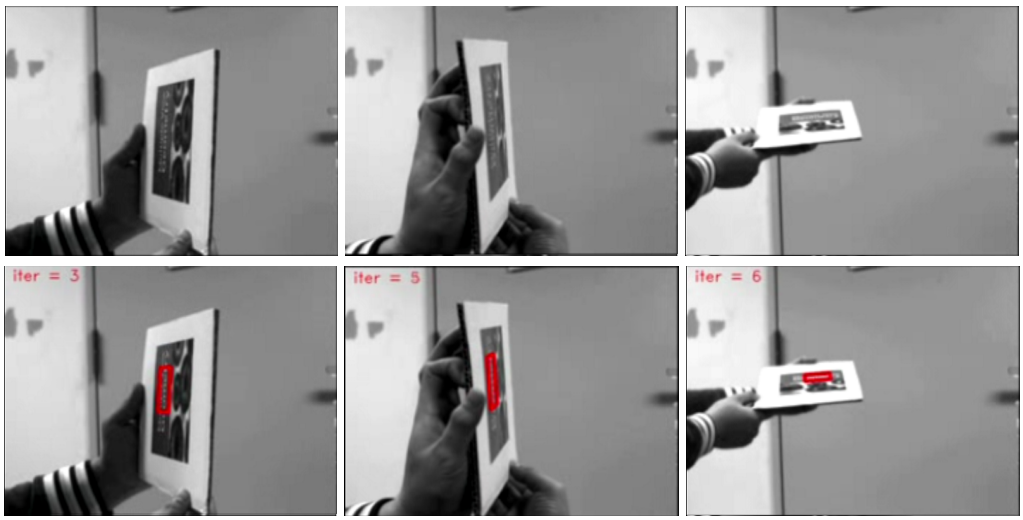


Fig. 3.14: Fig.3.12 と同様の実験を，画像を変えて行った結果．シャープなエッジだけで構成されている画像に対して，提案手法は特に有効である．

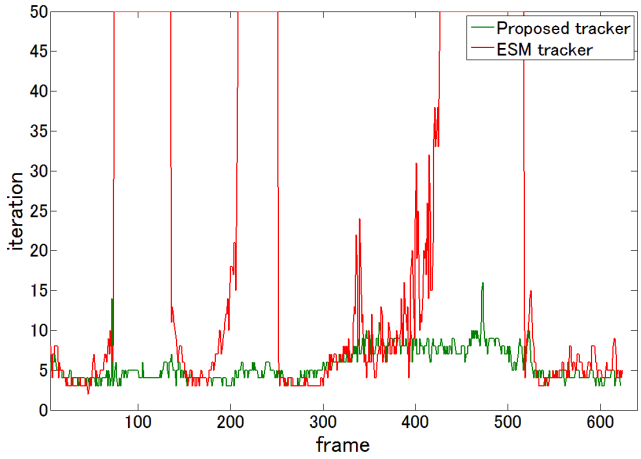


Fig. 3.15: Fig.3.14 の実験における，収束にかかった反復回数の比較．赤が解像度低下の影響を考慮していない ESM，緑が提案手法．従来手法で追跡困難だった箇所においても，少ない反復回数で解が収束している．

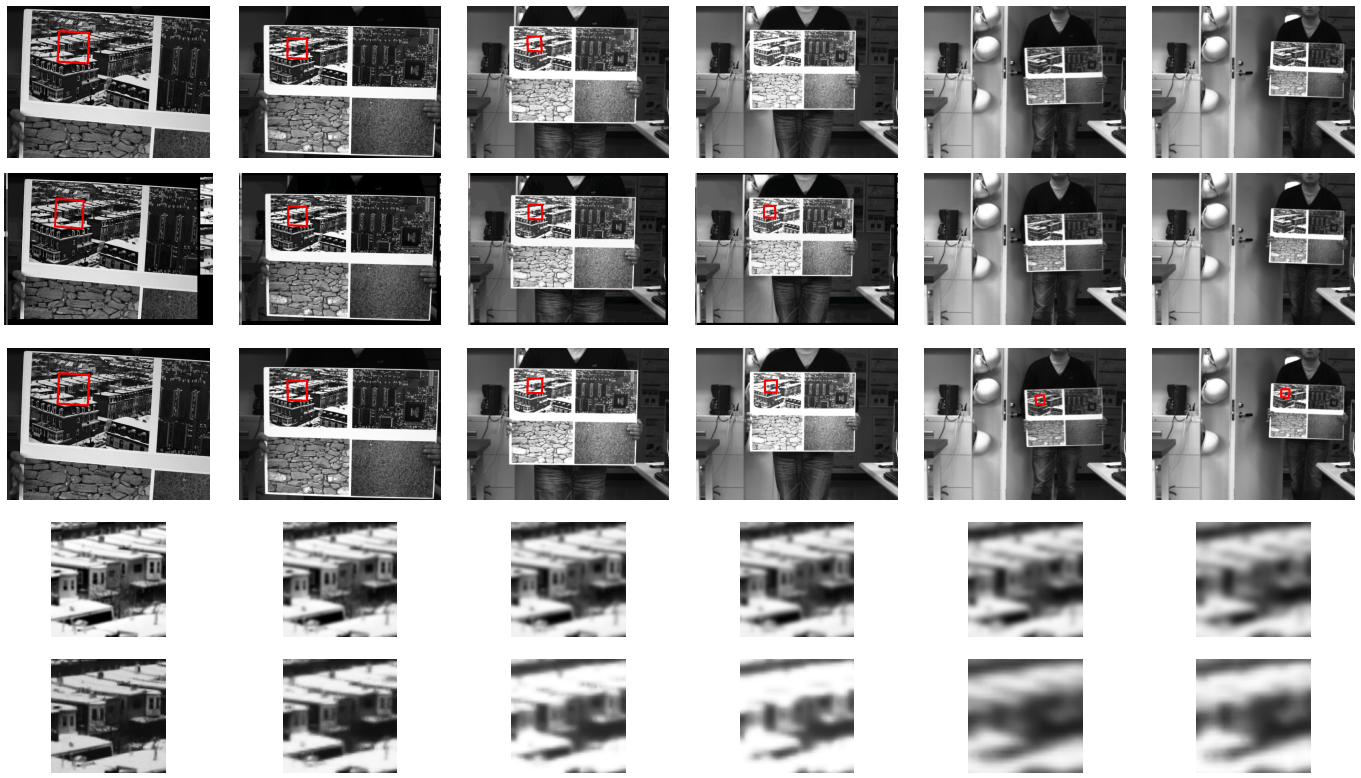


Fig. 3.16: 対象平面が徐々にカメラから遠ざかる場合の結果．最上段から，ESM の追跡結果，MI の結果，提案手法の結果，比較対象となる修正後のテンプレート，および撮影画像を変形したもの．各列は画像系列内の 1 フレームに対応する．

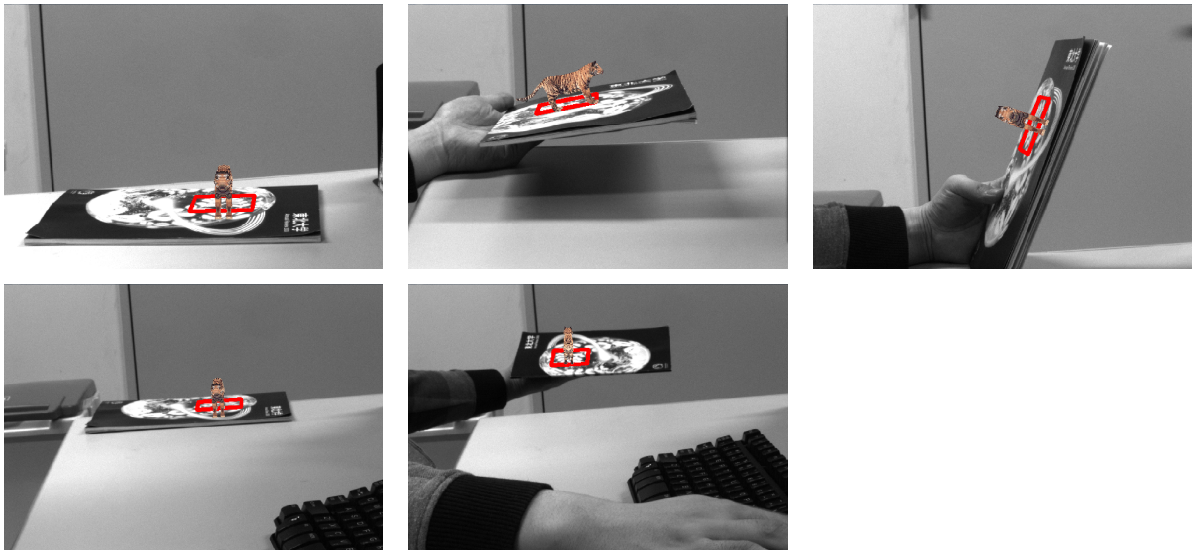


Fig. 3.17: AR のデモ．平面が相当傾く場合や，遠方にある場合でも安定して合成が可能.

3.9 直接法における多平面追跡

前節までは，単一平面の追跡モデルについて直接法を使って二次元射影変換 $\mathbf{H}$ を計算したが，平面の法線方向が既知という条件がなければ $\mathbf{R}, \mathbf{t}$ を一意に求めることはできない．一般的な SfM では法線ベクトルの値は未知であるため，平面同士の対応だけではカメラの位置と姿勢を復元できないのではないかと思うかもしれない．しかし，$\mathbf{R}, \mathbf{t}, \mathbf{n}$ をパラメータとして二次元射影変換を厳密に定義し，2 つ以上の異なる平面を追跡することで，これらを同時に推定できることが報告されている [32, 43, 58]．ここでは直接法における単一平面追跡のモデルを拡張し，複数平面の追跡をどのように行えば良いかについて解説する．

3.10 なぜ多平面なのか

特徴点ベースの方法では，多数の観測点と再投影された三次元点の誤差を最小化するようなパラメータ推定をすることで精度の向上を図っている．しかし，特徴点の対応づけとパラメータの推定はそれぞれ独立しているため，どんなにパラメータ推定の方法が優れていたとしても観測誤差自体を消すことはできない．一方，複数平面の直接法では観測（ここでは各ピクセルの濃淡値）とパラメータの推定が分離されておらず，そのまま最適化問題を解くことができる．ここで注意なのが，複数平面の直接法はテンプレートとなる画像と入力される現在の画像の 2 つのみから $\mathbf{R}, \mathbf{t}, \mathbf{n}$ を毎回推定しているので，全フレームを通した全体最適化ではなく，あくまで 2 フレーム間でのみでの最適化になっている．さらに，追跡する対象が平面であるという強い拘束をかけているため，空間上に異なる平面が多数配置されているシーンでのみ利用可能である．仮に空間上に平面が多数配置されていて，なおかつ画像間でシーンの変動が少ない場合（高フレームレートのカメラで撮影，またはゆっくりとした移動で撮影する等），単一平面の直接法ではできなかったシーンの形状とカメラ移動の同時推定が高精度にできるため，SfM への応用が期待される．

3.11 $\mathbb{SE}(3)$

直接法を複数平面のモデルに拡張するには，空間上に配置されたすべての平面の位置関係が剛体であり，同じカメラモーションに基づいていると仮定して解く必要がある．つまり，Fig.3.18 のように平面ごとに独立したパラメータを法線ベクトル $\mathbf{n}_i$ とし，$\mathbf{R}, \mathbf{t}$ が共通になるように式を変化させる必要がある．平面までの距離 d は $\mathbf{n}$ の内部に含まれているも

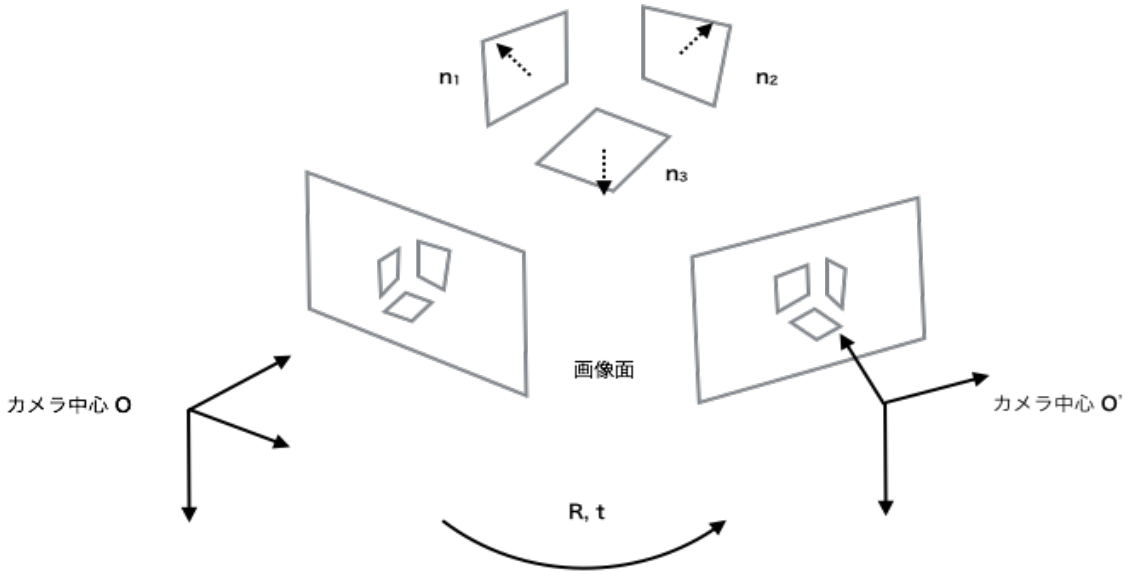


Fig. 3.18: 複数平面のモデル.

のとする．これを二次元射影変換の行列に直すと各平面ごとに

$$\mathbf{H} \quad = \quad \mathbf{K}(\mathbf{R} + \mathbf{t}\mathbf{n}_i^T)\mathbf{K}^{-1} \tag{3.17}$$

となる．ここで気になるのが，どのようにしてこれらのパラメータを簡潔に最小の形で表現するかである．回転行列の表現方法は様々あり

- ロール・ピッチ・ヨーそれぞれの軸を順に回転させていく方法.
- 回転軸角度表現.
- クォータニオン.

などが挙げられるが，ここでは後々の微分計算を単純化するために回転軸角度表現に基づく方法がとられている．この回転行列の表現を使って，平行移動ベクトルもひとまとめにした (4x4) 行列を **T** とすると

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.18}$$

と表現され，これは特殊ユークリッド群 $\mathbb{SE}(3)$ と呼ばれている．特殊線形群の時と同様に，特殊ユークリッド群もリー代数の基底 $\mathbf{A}_i$ の指数写像で表すことができる.

$$\mathbf{A}(\mathbf{x}) \quad = \quad \sum_{i=1}^{6} x_i \mathbf{A}_i \tag{3.19}$$

$$\mathbf{T}(\mathbf{x}) \quad = \quad exp(\mathbf{A}(\mathbf{x})) \tag{3.20}$$

以下に特殊ユークリッド群のリー代数の基底を列挙する．

$$\mathbf{A}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{A}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{A}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{A}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{A}_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{A}_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

各基底にかかる x_i がそれぞれ求めるべき回転と平行移動のパラメータになっており，x_1 から x_3 までが平行移動成分，x_4 から x_6 までが回転成分を表現している．したがって，m 個の平面を追跡する際に必要なパラメータは，特殊ユークリッド群の自由度+スケールを含む法線の自由度（ただし，どれか一つの平面にスケールを固定する必要があるので，一つ目の平面の自由度を 2 とする）の $6 + 3(m-1) + 2$ となり，二次元射影変換は $\mathbf{H}(\mathbf{T}(\mathbf{x}), \mathbf{n}_i)$ のような関数で表すことができる．

3.12 多平面モデルの最適化の式

問題を単純化するために一つの平面の式をみることにする．まず，最小化すべき評価関数は Eq.(1.27) の $\mathbf{H}$ を $\mathbb{SE}(3)$ の関数で置き換えたものとなっており

$$\arg\min_{\mathbf{x},\mathbf{n}} \sum_i \left[\mathcal{I}(w(\mathbf{p}_i^*; \mathbf{H}(\mathbf{TT}(\mathbf{x}), \hat{\mathbf{n}} + \mathbf{n}))) - \mathcal{I}^*(\mathbf{p}_i^*)\right]^2 \tag{3.21}$$

である．$\mathbf{T}$ と $\hat{\mathbf{n}}$ は反復計算の際に使う初期値となっている．合成関数の微分をするときに扱いやすい形にするため，上の式を変形すると

$$\arg\min_{\mathbf{x},\mathbf{n}} \sum_i \left[\mathcal{I}(w(\mathbf{p}_i^*; \mathbf{H}^{-1}(\mathbf{T}, \hat{\mathbf{n}})\mathbf{H}(\mathbf{TT}(\mathbf{x}), \hat{\mathbf{n}} + \mathbf{n})); w(\mathbf{H}(\mathbf{T}), \hat{\mathbf{n}})) - \mathcal{I}^*(\mathbf{p}_i^*)\right]^2 \tag{3.22}$$

となる．これを Forward Compositional 法を使ってまとめると以下の式が導かれる．

$$\mathbf{J}(\mathbf{0}) \begin{bmatrix} \mathbf{x} \\ \mathbf{n} \end{bmatrix} = -\mathbf{y}(\mathbf{0}). \tag{3.23}$$

$$\begin{aligned} \mathbf{J}(\mathbf{0}) &= \begin{bmatrix} \mathbf{J_x}(\mathbf{0}) & \mathbf{J_n}(\mathbf{0}) \end{bmatrix} \\ \mathbf{J_x}(\mathbf{0}) &= \mathbf{J}_I \mathbf{J}_w \mathbf{J_T} \mathbf{J_X}(\mathbf{0}) \\ \mathbf{J_n}(\mathbf{0}) &= \mathbf{J}_I \mathbf{J}_w \mathbf{J_N}(\mathbf{0}) \end{aligned}$$

$\mathbf{J}(\mathbf{0})$ はヤコビ行列，$\mathbf{y}(\mathbf{0})$ は残差ベクトルである．このようにして，一つの平面に関する線形連立方程式が得られたので，次に m 個の平面の最適化の式に並べかえると

$$\begin{bmatrix} \mathbf{J}_{1\mathbf{x}}(\mathbf{0}) & \mathbf{J}_{1\mathbf{n}}(\mathbf{0}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{2\mathbf{x}}(\mathbf{0}) & \mathbf{0} & \mathbf{J}_{2\mathbf{n}}(\mathbf{0}) & \mathbf{0} & \mathbf{0} \\ \vdots & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{J}_{m\mathbf{x}}(\mathbf{0}) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{m\mathbf{x}}(\mathbf{0}) \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{n}_1 \\ \vdots \\ \mathbf{n}_m \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1(\mathbf{0}) \\ \mathbf{y}_2(\mathbf{0}) \\ \vdots \\ \mathbf{y}_m(\mathbf{0}) \end{bmatrix} \tag{3.24}$$

となる．この大規模な線形連立方程式を解くことによって各平面の微小な変動量である $\mathbf{x}$ と $\mathbf{n}$ を推定することが可能であり，以下のような更新規則で反復計算をすれば良い．

$$\mathbf{T} \leftarrow \mathbf{TT}(\mathbf{x}) \tag{3.25}$$

$$\hat{\mathbf{n}} \leftarrow \hat{\mathbf{n}} + \mathbf{n} \tag{3.26}$$

3.13 シミュレーション画像による実験

複数平面追跡モデルの挙動を確認するために，シミュレーション画像による実験を行った．実験条件として，空間上で直角になるような 3 つの平面を配置し，手動で選択した任意の 6 点を中心とする (60×60) ピクセルのパッチを約 160 フレームに渡って追跡を行なった (Fig.3.19)．シミュレーション画像のサイズは (640×480) ピクセルとなっており，各平面には特定の軸周りに回転するような変形を与えて往復運動させている．最初の画像に与える初期値は，$\mathbf{T} = \mathbf{I}, \hat{\mathbf{n}} = (0, 0, 1)^T$ とし，法線方向に関しては真値から外れた適当な初期値になっている．最適化計算をするときに全ての法線の自由度を 3 にすると全体のスケールが定まらないため，$||\hat{\mathbf{n}}_1 + \mathbf{n}_1||^2 = 1$ となるように 1 個目の平面に対して拘束条件をかけている．つまり，カメラの初期位置から 1 個目の平面に下ろした垂線の長さが 1 となるような条件になっている．このようにしているのは，距離に関する事前知識がない状態で，画像のみが与えられていた場合に相対的なスケールしか復元できないからである．加えて，$\mathbf{T}$ の平行移動成分である $\mathbf{t}$ の大きさが 0.25 以下のときは法線を推定せずに平面の追跡を行うような処理をしている．これは $\mathbf{t} = \mathbf{0}$ のときに $\mathbf{J_n}(\mathbf{0})$ の rank が落ちて不定解になるためである．直感的にもわかりやすく，視差がないときには平面の変形が得られず法線を推定するための材料がなくなってしまうからである．

Fig.3.19 は上から 1,30,60,100,140 フレームにおける追跡結果と復元されたカメラ軌跡であり，右側の画像の赤い矩形領域が追跡された平面，緑のラインが法線ベクトル（見やすいように向きを反転し，長さを揃えた）になっている．$\mathbf{n}$ の初期値に適当な値を与えてるため，最初の数十フレームにおいては不安定な推定結果になってるが，法線の推定結果が安定したのちは滑らかなカメラ軌道を描いている．100 から 140 フレームは往復するような運動になっているが，その間のカメラ軌道も正しく重なっていることがわかる．Fig.3.20 は，160 フレーム時点での復元された平面同士の空間配置を 3 方向から撮影したものである．空間上で同一平面上にある 2 つの矩形領域の法線方向はほぼ一致しており，他の 2 平面とも直交関係が保たれていることがわかる．

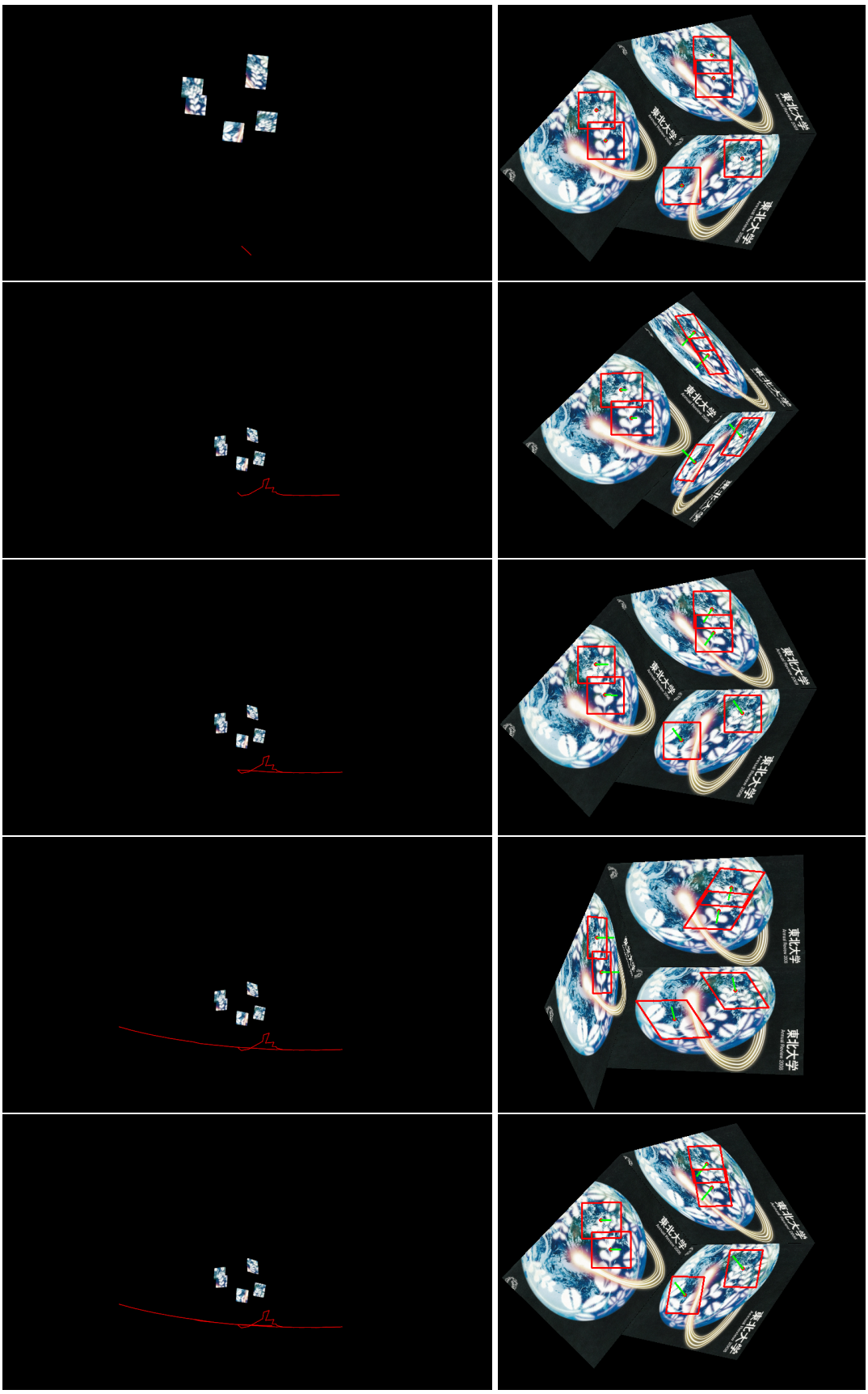


Fig. 3.19: シミュレーションによる複数平面の追跡.

Fig. 3.20: 推定された平面の空間配置.

第4章　直接法の点・直線混合による拡張

この章では，点追跡モデルの直接法を拡張して，直線も同時に最適化する手法について述べる．近年，直接法による画像追跡の研究がめざましく進歩しており，点，単一平面，複数平面モデルなど様々な方法が提案されてきた．点モデルは，複数平面モデルの問題点であった一般的な環境における SfM への扱いづらさを克服し，応用範囲を広げることが可能になったが，2 枚の画像しか与えられていない状況において，カメラモーションと空間の形状の推定を同時最適化の枠組みで解くことが不可能になっていた．本研究は一般的な SfM への応用を考慮し，点と直線要素を組み合わせた扱いやすい最適化手法を提案する．そして，点・直線混合モデルによって直線部分の空間的な位置推定をトラッキングと同時に推定できることを示す．

4.1　直接法における点追跡のモデル

前章では複数平面の追跡による SfM について論じたが，これはあらかじめ空間上に平面が配置されているという強い拘束条件が必要であった．この拘束条件は未知の環境では成り立つとは限らないため，実環境への応用の大きな障害となっている．一方，特徴点ベースの方法を使った SfM では，対応点を求めてから幾何学的なパラメータを推定するという 2 段階の手順を踏んでいるため，複雑な形状をした環境でもロバストに動くことが分かっている．では，限りなく平面を小さくして点として近似してしまうのはどうであろうか．これは Lucas-Kanade 法を使ったオプティカルフロー [39] に近い考え方であるが，この方法ではそれぞれの追跡領域を完全に独立したパラメータで置いているため，**R** と **t** を共通とする方法とは根本的に異なっている．

二次元射影変換行列を計算するには，同一平面上にある 4 つ以上の点がわかっていれば良いので，例として，最小構成である (2×2) ピクセルの微小な平面で複雑な 3 次元形状を近似し追跡する方法を考える．多数の平面は共通する **R** と **t** を持ち，さらに各平面に独立したパラメータである法線 **n** を持っている．ここで問題となるのが，4 つの点だけで法線を推定することになるということである．直接法の推定精度は，追跡するピクセルの数に大きく依存しており，少ない点で精度良く安定してパラメータを推定することは困難であ

る．そして計算量の面でも，独立した $\mathbf{n}$ を大量に計算しなければならず得策とは言えない．仮にすべての法線が既知で回転と平行移動成分だけ推定する場合には，全てのピクセルが同じ拘束式に乗るため，当然ではあるが多平面の追跡の安定性は向上する．以上のように，複数の平面で形状を近似して全てのパラメータを直接法によって推定する方法は良いとは言えないが，近年の研究によって，直接法の点追跡モデルとエピポーラ幾何による深度推定をうまく組み合わせることで，複雑な形状においてもリアルタイムに SfM ができることが報告されている [18, 10, 17]．次節からは，直接法の点追跡モデルについて解説を行なっていく．

4.2 直接法の一般化モデルである点追跡の利点

点追跡モデルを使った直接法では，カメラの原点から点までの深度 d とカメラの移動 $\mathbf{T} \in \mathbb{SE}(3)$ を未知のパラメータとして推定をしている．そして，トラッキングとエピポーラ幾何による深度推定に分離して，計算量の削減と安定化の工夫を行なっている．つまり，多数の点の深度を最適化の式に含めず独立して計算することで，多平面の追跡で起こるような不安定なパラメータの挙動を抑えることができる．トラッキングでは直接法で $\mathbf{T}$ のみを，エピポーラ幾何ではトラッキングで得られた $\mathbf{T}$ をもとに各点の d を計算している．こうすることで，トラッキングの最中は全点群に共通したパラメータのみで問題を解くことができ，モーションブラーに頑健で安定した $\mathbf{T}$ の推定が可能になる．特徴点ベースの方法では，局所的な点の情報だけで特徴点抽出を行なっているため，モーションブラーに弱いという短所がある．直接法の点追跡モデルの最大の利点は，シーンの形状の事前知識が必要ないという点にあり，これによって様々な環境で SfM をする際に直接法を適用することが可能になっている．

4.3 点追跡モデルの計算処理の概要

直接法の点追跡モデルの処理手順は以下の通りである．

- **初期化** キーフレームから追跡するピクセル $\mathbf{p}_i$ を選択し，深度 d_i とその分散 V_i の初期化をする．
- **トラッキング** 深度 d_i を固定して，対象形状が剛体という仮定のもとに $\mathbf{T}(\mathbf{x})$ を推定する．深度の分散が最適化計算の重み付けに使われる．推定された $\mathbf{x}$ をもとに $\mathbf{T} \leftarrow \mathbf{T}(\mathbf{x})\mathbf{T}$ でパラメータを更新していく．

- **深度の推定** トラッキングで得られた $\mathbf{T}$ を使ってエピポーラ線をひき，線上を探索することで観測される深度 $d_{obs(i)}$ と分散 $V_{obs(i)}$ の推定を行う．その後，観測値と事前に持っていた値を混合して深度と分散を更新する．

- **キーフレームの更新** キーフレームからのカメラ移動量が大きくなった場合にキーフレームを更新し，一つ前のキーフレームで推定された d_i と V_i を使って初期化を行う．

Engel ら [18] は追跡するピクセルを選択するときに Semi-Dense の方法を取っている．これは画像上にあらわれるエッジの強度が高い部分を追跡対象のピクセルとして選び，それ以外のテクスチャが少ないところに関しては無視して計算コストを下げる方法である．次の節で詳細について説明するが，トラッキングの最適化の式には画像の空間微分の成分が入っているため，テクスチャが少なく特徴がないピクセルに関しては $\mathbf{T}$ の推定にほとんど寄与せず，無視してしまっても問題はないといえる．

4.4 トラッキングの式の導出

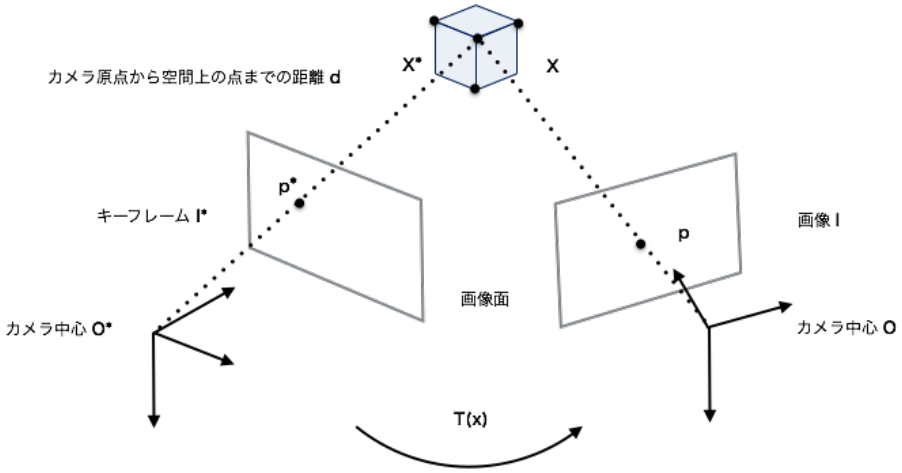


Fig. 4.1: 直接法の点追跡モデル．

最適化の式を導出過程について説明をしていく．参照元となるキーフレームの画像を $\mathcal{I}^*$，カメラが移動したときに入力される画像を $\mathcal{I}$，キーフレームにおいて追跡したいピクセルを $\mathbf{p}_i^* = (u_i^*, v_i^*, 1)^T$，入力画像上でそれに対応するピクセルを $\mathbf{p}_i$ と置く．画像上で対応する点の濃淡が不変であると仮定すると

$$\mathcal{I}^*(\mathbf{p}_i^*) = \mathcal{I}(\mathbf{p}_i) \tag{4.1}$$

が成り立つ．$\mathbb{SE}(3)$ のパラメータを $\mathbf{T(x)}$，その回転・平行移動成分を $\mathbf{R_x}, \mathbf{t_x}$ とすると，キーフレーム画像を基準とした空間上の点 $\mathbf{X}_i^*$ と $\mathbf{p}_i$ の関係は

$$\mathbf{p}_i \propto \mathbf{K}(\mathbf{R_x}\mathbf{X}_i^* + \mathbf{t_x}) \tag{4.2}$$

となる．ここで $\propto$ を関数 π の形に書き換え，キーフレームのカメラ原点から空間上の点までの距離を d_i と置いて $\mathbf{X}_i^*$ を変形すると以下のようになる．

$$\begin{aligned}
\pi\begin{pmatrix} x \\ y \\ z \end{pmatrix} &= \begin{pmatrix} x/z \\ y/z \end{pmatrix}. \\
\mathbf{X}_i^* &= \mathbf{K}^{-1} d_i \mathbf{p}_i^*. \\
\mathbf{p}_i &= \pi(\mathbf{K}(\mathbf{R_x}\mathbf{K}^{-1} d_i \mathbf{p}_i^* + \mathbf{t_x})).
\end{aligned} \tag{4.3}$$

簡潔に表記するため，$\mathbf{p}_i = \mathbf{w}(\mathbf{p}_i^*, d_i, \mathbf{K}, \mathbf{T(x)T})$ のように反復計算用の初期値も組み込んだ $\mathbf{w}$ の関数の形にし，Eq.(4.1) に代入すると最適化する評価関数は

$$\arg\min_x \sum_i (\mathcal{I}(\mathbf{w}(\mathbf{p}_i^*, d_i, \mathbf{K}, \mathbf{T(x)T})) - \mathcal{I}^*(\mathbf{p}_i^*))^2 \tag{4.4}$$

となる．ここで注意すべきところは，$\mathbf{x}$ 以外全て定数という点である．この式を平面追跡の時と同様に Forward Compositional 法を使って計算していくと

$$\begin{aligned}
\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} &= \mathbf{R}\mathbf{K}^{-1} d_i \mathbf{p}_i^* + \mathbf{t}. \\
\mathbf{J}_{row(i)} &= \frac{1}{z_i}\begin{pmatrix} \nabla\mathcal{I}_u f_x & \nabla\mathcal{I}_v f_y \end{pmatrix}
\begin{pmatrix} 1 & 0 & -\frac{x_i}{z_i} & -\frac{x_i y_i}{z_i} & (z_i + \frac{x_i^2}{z_i}) & -y_i \\ 0 & 1 & -\frac{y_i}{z_i} & -(z_i + \frac{y_i^2}{z_i}) & \frac{x_i y_i}{z_i} & x_i \end{pmatrix}. \\
y_i(\mathbf{0}) &= \mathcal{I}(\mathbf{w}(\mathbf{p}_i^*, d_i, \mathbf{K}, \mathbf{T(0)T})) - \mathcal{I}^*(\mathbf{p}_i^*). \\
\mathbf{x} &= -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}\mathbf{y}(\mathbf{0}).
\end{aligned} \tag{4.5}$$

が導かれる．求められた $\mathbf{x}$ によって 1 反復ごとの更新規則は以下のようになる．

$$\mathbf{T} \leftarrow \mathbf{T(x)T}. \tag{4.6}$$

4.5 エピポーラ幾何による深度の推定

Fig.4.2 のように，深度推定では探索範囲 ($d_{min} < d < d_{max}$) を決め，Eq.(4.3) に代入してエピポーラ線上で点の対応を探していく．点同士の類似度を計算する方法は SSD を使っ

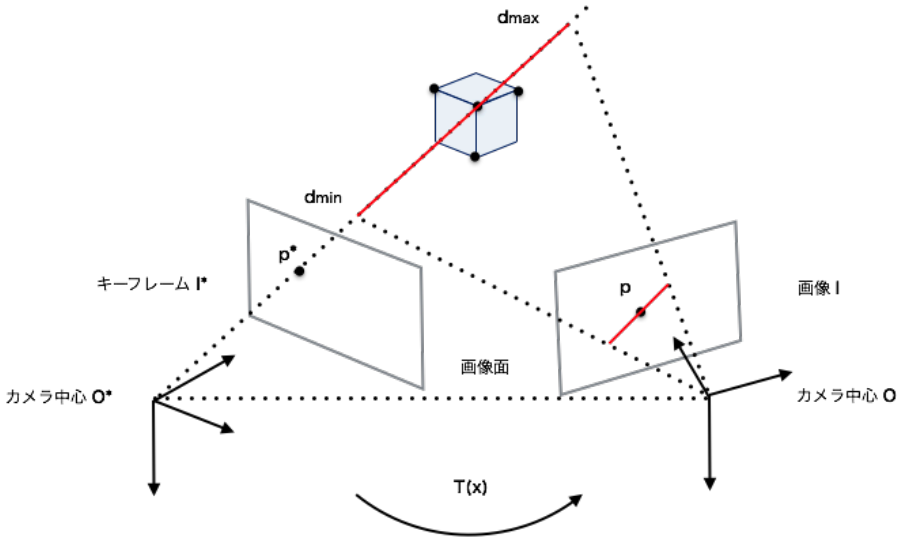


Fig. 4.2: エピポーラ幾何による深度の推定.

ている．具体的には，エピポーラ線に沿う 1 ピクセル間隔に配置した 5 つの点をスライドさせていき，SSD の応答が最も小さくなる点を対応点としている．そしてサブピクセル精度になるように補正を行った後に，三角測量で観測深度 d_{obs} を計算している．

観測深度の分散 V_{obs} の設計では，分散が主に 3 つの要素で成り立っていると仮定し以下のようにしている．

- **対応点の判別しやすさから影響する分散 σ_1^2**．SSD で使う 5 点の微分値の平均 g_p が高いほど点同士マッチングの判別性能が上がるため，$1/g_p^2$ を分散としている．
- **エピポーラ線の方向と画像中のエッジの方向から影響する分散 σ_2^2**．エピポーラ線の向き $\mathbf{l}_{dir}$ と画像中のエッジの向き $\mathbf{g}_{dir}$ が被った場合，エッジ上を探索することになるので誤差の原因になる．そのため，それらを内積して 2 乗し，逆数をとったものを分散としている．
- **画像上の視差から深度への変換 α**．求めたいのは深度の分散なので，画像座標から深度への変換を行う係数になっている．画像上でエピポーラ探索するピクセルの長さを δ_l，探索する深度の範囲の長さを δ_d とすると，$\alpha = \delta_d/\delta_l$ になっている．

この 3 つの要素をまとめると

$$V_{obs} = \alpha^2(\sigma_1^2 + \sigma_2^2) \tag{4.7}$$

となる．

以上のようにして求まった深度と分散を，これまでの推定値に混合することでパラメータを更新している．

$$d = \frac{Vd_{obs} + V_{obs}d}{V + V_{obs}}. \tag{4.8}$$

$$V = \frac{VV_{obs}}{V + V_{obs}}. \tag{4.9}$$

4.6 TUMデータセットでの実験

直接法の点追跡モデルの性能を確かめるため，RGB-D カメラ用の SfM を評価するために公開された TUM データセット [62] を使用した．実験条件として，kinect で撮影された (640×480) の RGB 画像のみを使っている．フレームレートは 30fps であり，一般に普及してるカメラとほぼ同等である．最初のキーフレームにおける深度 d の初期化には特徴点ベースの方法を用いており，5 点アルゴリズムで求めた $\mathbf{R}, \mathbf{t}$ でエピポーラ幾何による深度推定を行った．このようにした理由は，d が不正確な場合にトラッキングの精度が落ち，不正確な $\mathbf{R}, \mathbf{t}$ が出力されてしまうからである．トラッキングの計算では，coarse-to-fine と呼ばれる手法を適用した．これはコンピュータビジョンでよく使われる方法であり，処理する画像を荒いものから開始し，徐々に鮮明なものに切り替えることでパラメータの収束半径を広げることが可能である．この実験では (320×240) の画像を初期入力とした．最適化計算の重み付けでは，外れ値に頑健になるように Huber 関数と深度の分散 V_i の逆数を掛け合わせたものを使用した．

Fig.4.3 と Fig.4.4 が処理結果である．画像中の赤い点が追跡中のピクセルで，それに対応したカメラ軌道と空間上の 3 次元点が表示されてある．ただしこの 3 次元点は，一つ前のキーフレームで推定された点（外れ値と判定された点は除外した）のみになっている．この結果を見ると，画像中のエッジなどの点群が人の目にもわかりやすい形で復元されており，カメラが一周したときに推定されたカメラ軌道のループが閉じていることから，正しくパラメータの推定が行われたことがわかる．特徴点ベースの方法では疎な対応点から SfM するため，計算量の多い後処理で密な対応をつけなければ形状がわかりにくいことが難点である．

実験に用いた画像はテーブルの上に乱雑に物が置かれ，テクスチャが十分にある平面が少ない状況になっている．そのため多平面追跡モデルでは扱い辛い問題であったが，点追跡モデルを適用することで直接法でも複雑な環境下で SfM することが可能であることがわかった．

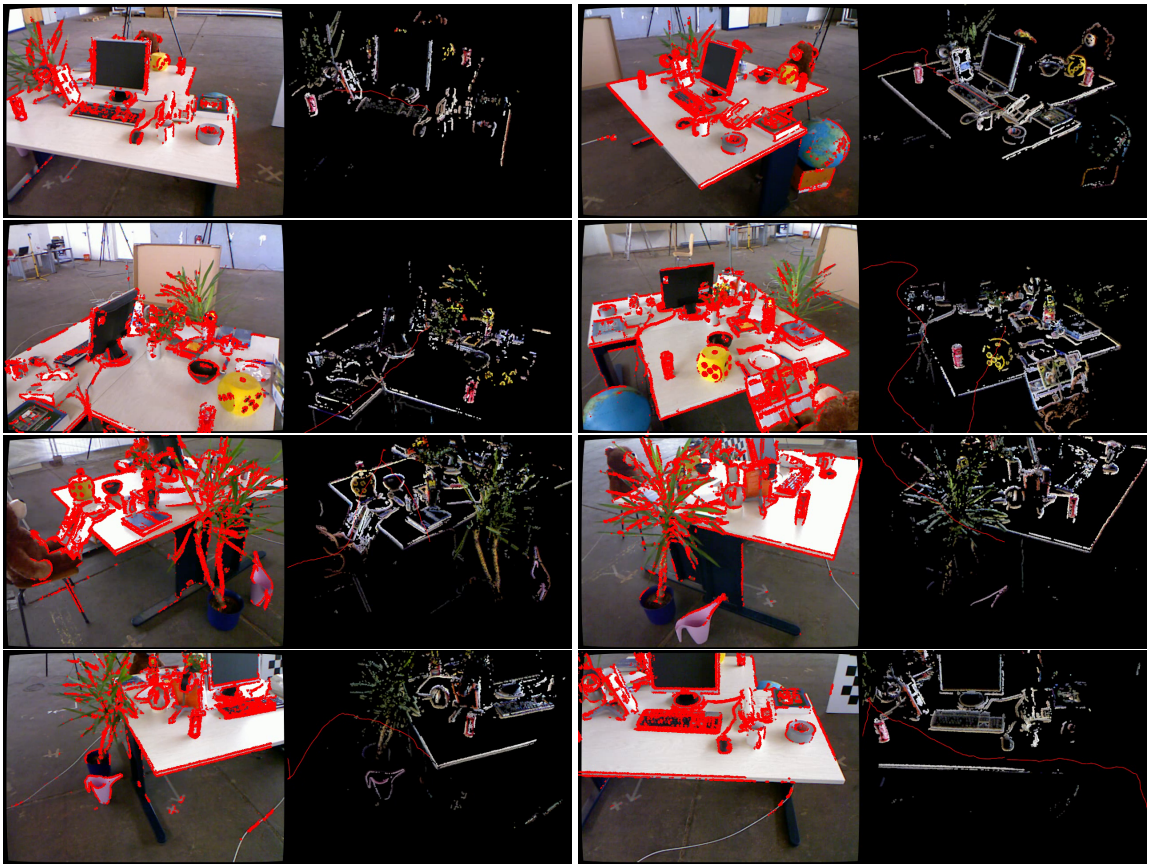


Fig. 4.3: TUM データセットの復元例.

4.7　瓦礫状の地形を移動するロボットへの応用

追加実験として瓦礫フィールド上で動く 4 脚を持つロボットにカメラを搭載し，SfM を行った．このロボットは腹ばい移動をしており，時間経過と共に足元の瓦礫が崩れ変動する状態になっている．直接法ではピクセルサイズが大きい場合に処理時間が大幅に増加するため，カメラから入力される (1920×1200) ピクセルのグレースケール画像を 1/4 に縮小したもので実験をした．

Fig.4.5 が実験結果である．このシークエンスではロボットが前進運動をしているため，エピポーラ線の向きが進行方向に固定されやすくなっている．観測される深度の分散は，エピポーラ線の向きとエッジの向きが直交してるほど小さくなるので，横方向のエッジの復元が多い結果となっている．ロボットが移動した際にカメラ前方の瓦礫が持ち上がり，剛体形状の仮定が崩れるシーンもあったが，形状が変化していない他の多数の点がカバーすることによって SfM が失敗せずに継続できることがわかった．

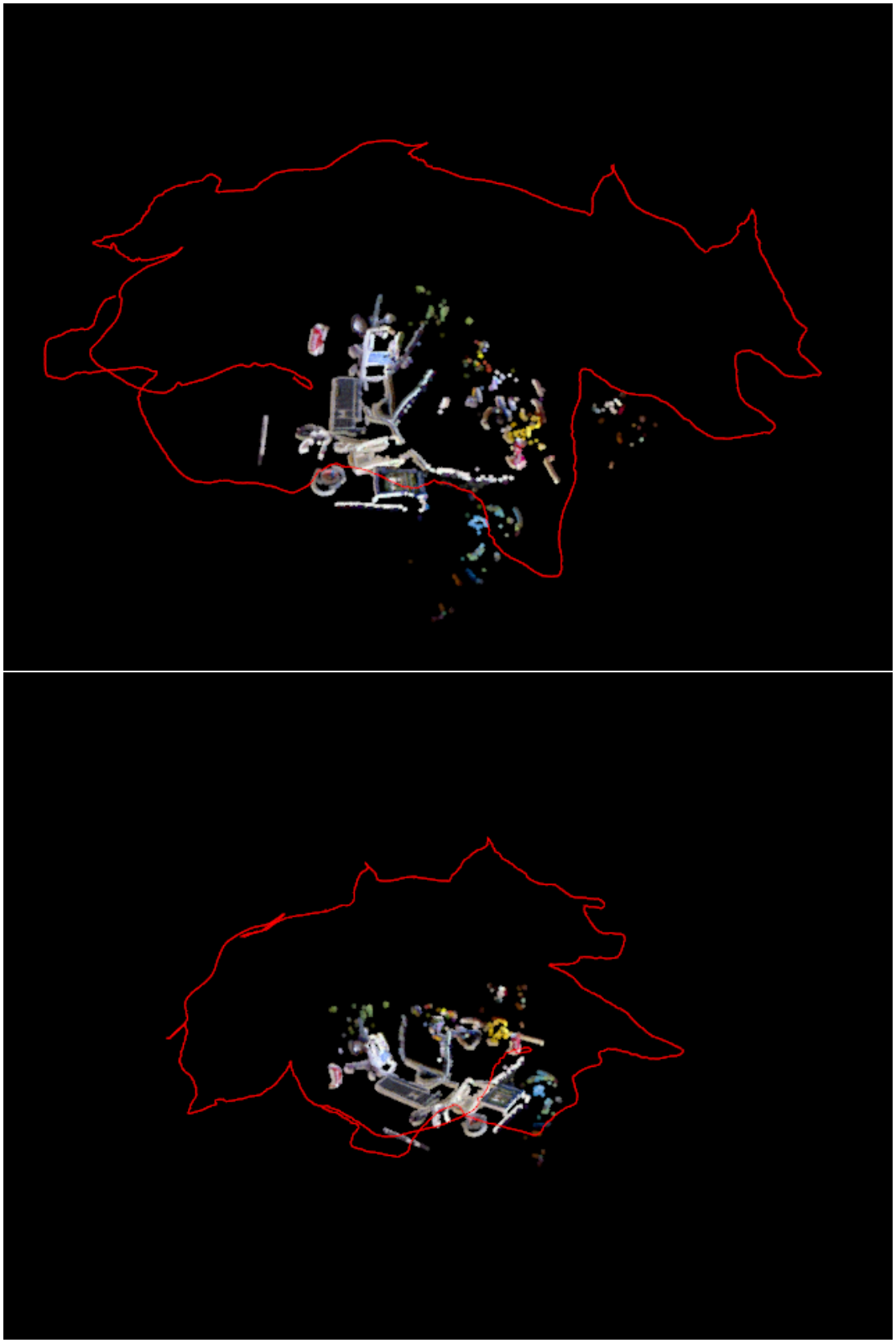


Fig. 4.4: 推定されたカメラ軌道.

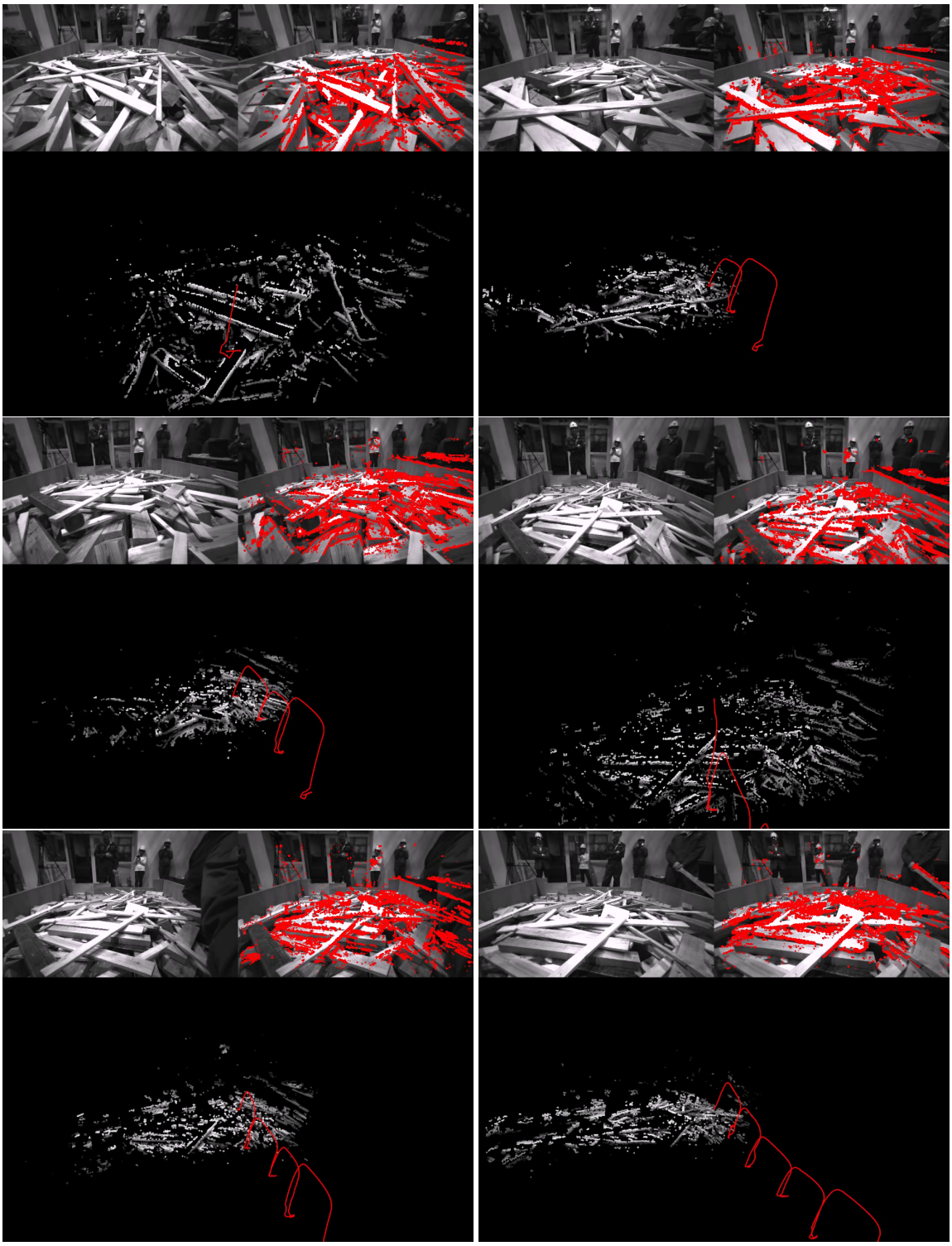


Fig. 4.5: 瓦礫状の地形を移動するロボット上のカメラを用いて行った SfM.

4.8 直線モデルへの拡張

最近の研究によって，直接法でシーンの形状とカメラモーションを推定する方法は様々提案されており，その分類は大きく 3 つに分かれる．1 つめは，各ピクセルに独立した深度パラメータを与え，複数フレームの情報を使いカメラモーションと同時に推定する方法 [47, 17] である．これは特徴点ベースの時の方法におけるバンドル調整のような役割をもっており，大規模な問題を同時最適化の枠組みで解くことができるが，シーンの形状についての仮定を何も置いておらず，画像枚数が少ない時には精度が落ちることが問題である．2 つめは，カメラモーションと深度を分割して推定する方法 [18, 10] である．この方法は，カメラモーションの推定には直接法を，深度推定にはエピポーラ幾何をつかっており，同時最適化という形にはなっていない．3 つめは，シーンの形状が多数の平面によって構成されていると仮定し，キーフーレムと現在の入力画像のみからカメラモーションと形状（ここでは各平面の法線）を推定する方法 [32, 58, 43] がある．これはシーンの形状に平面という強い拘束をいれているため，一般的な SfM では扱いにくい手法となっている．

直線モデルの考え方に最も近いのは 3 番目であるが，提案手法では直線という緩い拘束条件になっているため様々なシーンへの応用が可能となる．しかし，空間上で 3 本の線形独立な直線が存在しなければ，カメラ運動と形状を決めることができないという問題点がある．複数平面モデルのときも同様に，異なる 2 つ以上の平面を追跡しなければ解が求まらないという条件があり，安定してそれらの条件を満たすことは困難である．そのため，我々は 2 番の方法に直線モデルを加えた形で拡張し，多数の点群によって不良問題の発生を回避している．

直接法の分類と提案手法の比較について Table.4.1 にまとめる．

4.9 直線モデルの導出

この節では，直接法の直線モデルの最適化の式の流れについて説明をしていく．まず空間上に m 本の直線がある状況を想定し，その j 番目の直線 l_j において，キーフレームのカメラ原点から見たときの始点と終点の深度をそれぞれ α_j,β_j とする．l_j が画像上で n_j 個のピクセルを持ち，その i 番目の要素を p^*_{ji} と置き，それに対応する空間上の深度を，l_j を (1-s:s) に内分する点とすると

$$d_{ji} = (1 - s)\alpha_j + s\beta_j \tag{4.10}$$

Table 4.1: 直接法の分類

	[47, 17]	[18, 10]	[32, 58, 43]	提案手法
計算に必要な画像の最低枚数	複数 (2 <)	2	2	2
追跡するシーンの形状	自由	自由	複数平面が存在	自由（直線が存在しない場合は点追跡モデルに等しくなる）
カメラモーションと形状の推定	同時最適化	分割して推定	同時最適化	点は分割推定+直線部分は同時最適化

となる．それぞれの直線の位置関係が剛体の仮定を満たしており，カメラの動きが $\mathbf{T(x)} \in \mathbb{SE}(3)$ とすると，Eq.(4.3) から

$$\mathbf{p}_{ji} \quad = \quad \pi(\mathbf{K}(\mathbf{R}_\mathbf{x}\mathbf{K}^{-1}d_{ji}\mathbf{p}^*_{ji} + \mathbf{t}_\mathbf{x})). \tag{4.11}$$

が成り立つ．反復解法の初期値を $(\hat{\alpha}_j, \hat{\beta}_j, \mathbf{T})$ とすると最適化の評価関数は

$$d_{ji} = (1-s)(\hat{\alpha}_j + \alpha_j) + s(\hat{\beta}_j + \beta_j) \tag{4.12}$$

$$\arg\min_{\mathbf{x},\alpha,\beta} \sum_j \sum_i (\mathcal{I}(\mathbf{w}(\mathbf{p}^*_{ji}, d_{ji}, \mathbf{K}, \mathbf{T(x)T})) - \mathcal{I}^*(\mathbf{p}^*_{ji}))^2 \tag{4.13}$$

となり，直線モデルで求めるパラメータの個数は $6 + 2m$ である．この評価関数を点モデルの時と同様に，Forward Compositonal 法を使って解くと以下の式が導かれる．

$$\begin{pmatrix} x_{ji} \\ y_{ji} \\ z_{ji} \end{pmatrix} = \mathbf{R}\mathbf{K}^{-1}d_{ji}\mathbf{p}^*_{ji} + \mathbf{t}.$$

$$\mathbf{J}_{j\mathbf{x}}(row(i)) = \frac{1}{z_{ji}} \begin{pmatrix} \nabla\mathcal{I}_u f_x & \nabla\mathcal{I}_v f_y \end{pmatrix} \begin{pmatrix} 1 & 0 & -\frac{x_{ji}}{z_{ji}} & -\frac{x_{ji}y_{ji}}{z_{ji}} & (z_{ji} + \frac{x_{ji}^2}{z_{ji}}) & -y_{ji} \\ 0 & 1 & -\frac{y_{ji}}{z_{ji}} & -(z_{ji} + \frac{y_{ji}^2}{z_{ji}}) & \frac{x_{ji}y_{ji}}{z_{ji}} & x_{ji} \end{pmatrix}.$$

$$\mathbf{v}_{ji}1 = \mathbf{R}\mathbf{K}^{-1}(1-s)\mathbf{p}^*_{ji} + \mathbf{t}.$$

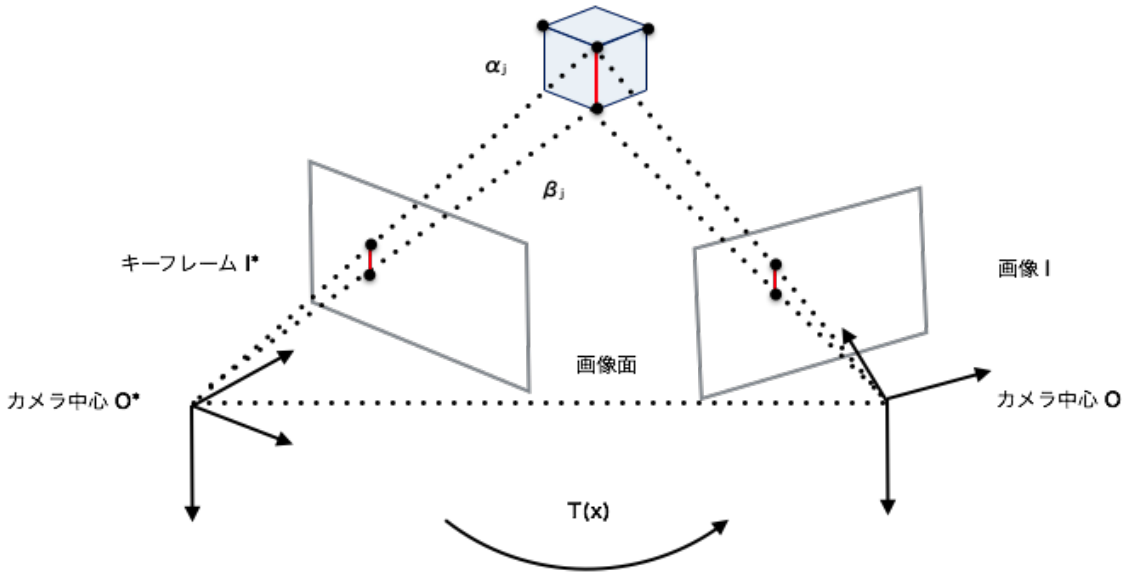


Fig. 4.6: 直線追跡のモデル.

$$\mathbf{v}_{ji}2 = \mathbf{R}\mathbf{K}^{-1}s\mathbf{p}^*_{ji} + \mathbf{t}.$$

$$\mathbf{J}_{j(\alpha,\beta)}(row(i)) = \frac{1}{z_{ji}}\begin{pmatrix}\nabla\mathcal{I}_u f_x & \nabla\mathcal{I}_v f_y\end{pmatrix}\begin{pmatrix}1 & 0 & -\frac{x_{ji}}{z_{ji}} \\ 0 & 1 & -\frac{y_{ji}}{z_{ji}}\end{pmatrix}\begin{pmatrix}\mathbf{v}_{ji}1 & \mathbf{v}_{ji}2\end{pmatrix}.$$

$$y_j(row(i)) = \mathcal{I}(\mathbf{w}(\mathbf{p}^*_{ji}, d_{ji}, \mathbf{K}, \mathbf{T}(\mathbf{0})\mathbf{T})) - \mathcal{I}^*(\mathbf{p}^*_{ji}).$$

$$\begin{bmatrix}\mathbf{J}_{1\mathbf{x}} & \mathbf{J}_{1(\alpha,\beta)} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{2\mathbf{x}} & \mathbf{0} & \mathbf{J}_{2(\alpha,\beta)} & \mathbf{0} & \mathbf{0} \\ \vdots & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{J}_{m\mathbf{x}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{m(\alpha,\beta)}\end{bmatrix}\begin{bmatrix}\mathbf{x} \\ \alpha_1 \\ \beta_1 \\ \vdots \\ \alpha_m \\ \beta_m\end{bmatrix} = -\begin{bmatrix}\mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_m\end{bmatrix} \tag{4.14}$$

注意点として，直線モデルを単独で使う場合はどれか一つの直線のスケールを固定する必要がある（l_1 の α を 1 にするなど）．以上で求められた $\mathbf{x}, \alpha, \beta$ によって，1 反復ごとの更新規則は以下のようになる．

$$\mathbf{T} \leftarrow \mathbf{T}(\mathbf{x})\mathbf{T}. \tag{4.15}$$

$$\hat{\alpha} \leftarrow \hat{\alpha} + \alpha. \tag{4.16}$$

$$\hat{\beta} \leftarrow \hat{\beta} + \beta. \tag{4.17}$$

直線モデルと点モデルの混合をするには，評価関数に Eq.(4.4) を加えるだけで良い．

$$\begin{aligned}\arg\min_{\mathbf{x},\alpha,\beta} &\sum_i (\mathcal{I}(\mathbf{w}(\mathbf{p}_i^*, d_i, \mathbf{K}, \mathbf{T}(\mathbf{x})\mathbf{T})) - \mathcal{I}^*(\mathbf{p}_i^*)) \\ &+ \sum_j \sum_i (\mathcal{I}(\mathbf{w}(\mathbf{p}_{ji}^*, d_{ji}, \mathbf{K}, \mathbf{T}(\mathbf{x})\mathbf{T})) - \mathcal{I}^*(\mathbf{p}_{ji}^*))^2 \end{aligned} \tag{4.18}$$

4.10 混合モデルの実験

TUM データセット [62] を用いて提案した手法の SfM を行なった．この実験では，(640×480) の画像を点・直線モデルで 30 フレーム追跡し，フレーム 1 とフレーム 30 の 2 枚の画像から推定された直線形状と 1-30 フレームで計算された点群の比較を行なっている．比較をわかりやすくするために，深度計算後の平滑化はしていない．キーフレームの深度の初期化には kinect の深度センサーから得られた値を使い，画像上の直線領域の抽出方法は確率的 Hough 変換を利用した．Fig.4.7 が実験結果である．上の画像の赤の点と緑の線はそれぞれ，点モデルで追跡された領域と直線モデルで追跡された領域になっており，下の画像が復元された 3 次元形状となっている．復元されたディスプレイの境界部分を見ると，点モデルで推定された領域はノイズが乗った荒い推定になっているが，直線モデルの方は直線の拘束式が入っているため綺麗な形状が出ていることがわかる．画像上で抽出された直線は横方向のものが多く，直線モデルのみの計算では不安定解に陥る可能性が高くなるが，点モデルと組み合わせて最適化することで安定した解が得られている．

次に，精度の評価をするために，同じデータセットに付随されているモーションキャプチャーから得られたカメラ軌跡との比較を行った．ここで，推定されたカメラ軌跡のスケールは各手法で異なっているので，モーションキャプチャーのカメラ軌跡に合うように相似変換をして，カメラ位置 $\mathbf{t}$ の絶対誤差を計算した．通常の計算では，このデータセットで最適化の計算に入力する点数は約 3 万点であるが，直線モデルの影響をより明確にするために，追跡する点の上限を 1 万点程度に抑えた厳しい条件で実験を行っている．点・直線混合モデルでは，全体の追跡点の 10-20 パーセントが直線の追跡に使われている．各追跡点における最適化計算の重み付けは，推定された深度の分散 V の逆数をかけたものになっており，直線モデルの部分に関しては例外的に重みを 1 に固定して計算をしている．そのため，直線モデルが推定精度に影響を及ぼす寄与率が，単純に 10-20 パーセントにはなっていないことに注意を払う必要がある．Fig.4.8 が精度評価の結果である．推定されたカメラ軌跡を見ると，従来手法ではフレームが進むに従って推定が激しく乱れているが，提案手法では安定して精度良く真値に近い推定値が得られていることがわかる．このような結果が得られた理由は，直線モデルの計算を最適化に加えることで，点モデルのみ

に比べて高精度に形状とカメラモーションの推定が可能になったからである．直接法は反復解法で初期値の近傍にある解を計算しているため，十分に近い初期値から計算をスタートしなければ精度を保つことが困難であり，ずれた推定値を使ってさらに計算すると大きな誤差につながってしまう．従来手法の点モデルは，点ごとに独立した形状のパラメータを持っているので，最適化計算に必要なパラメータ数が多く，推定された形状は外れ値を含む荒いものになっている．一方，提案手法では，直線部分の形状を少ないパラメータ数で表すことができ，さらにカメラモーションと同時に最適化の枠組みで解くことが可能であるため，高精度な推定結果が得られる．

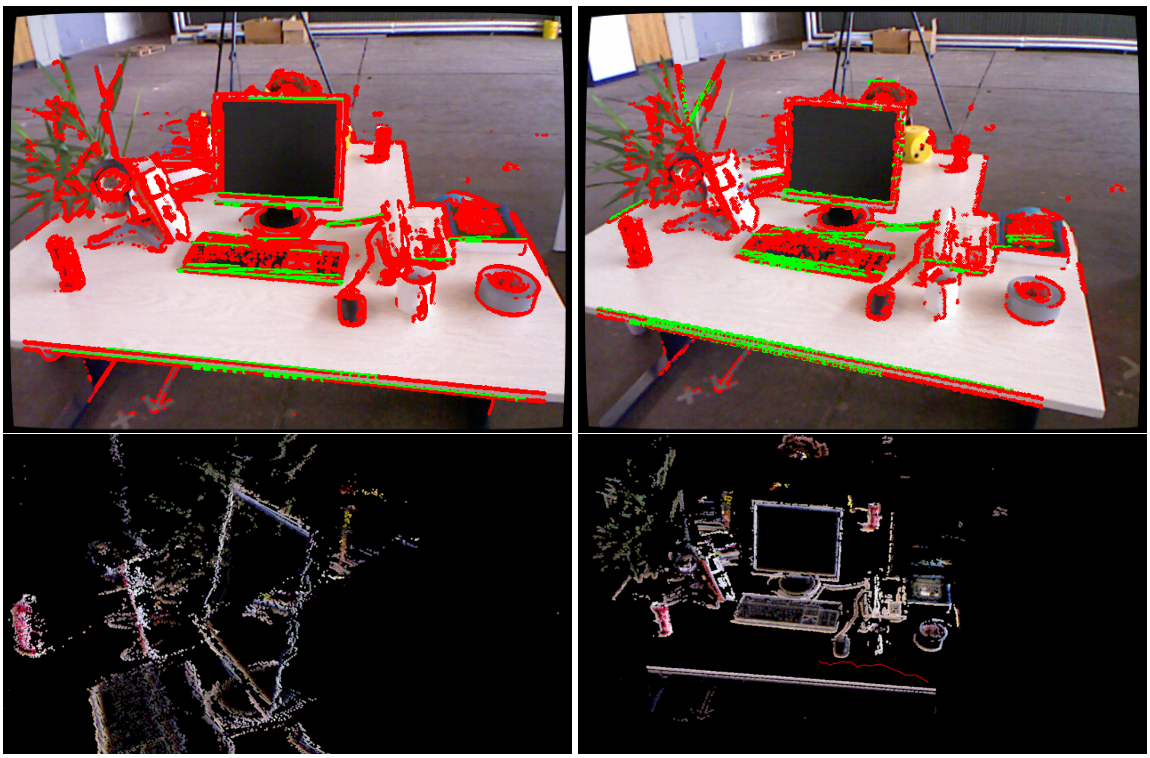


Fig. 4.7: 点・直線混合モデルの結果.

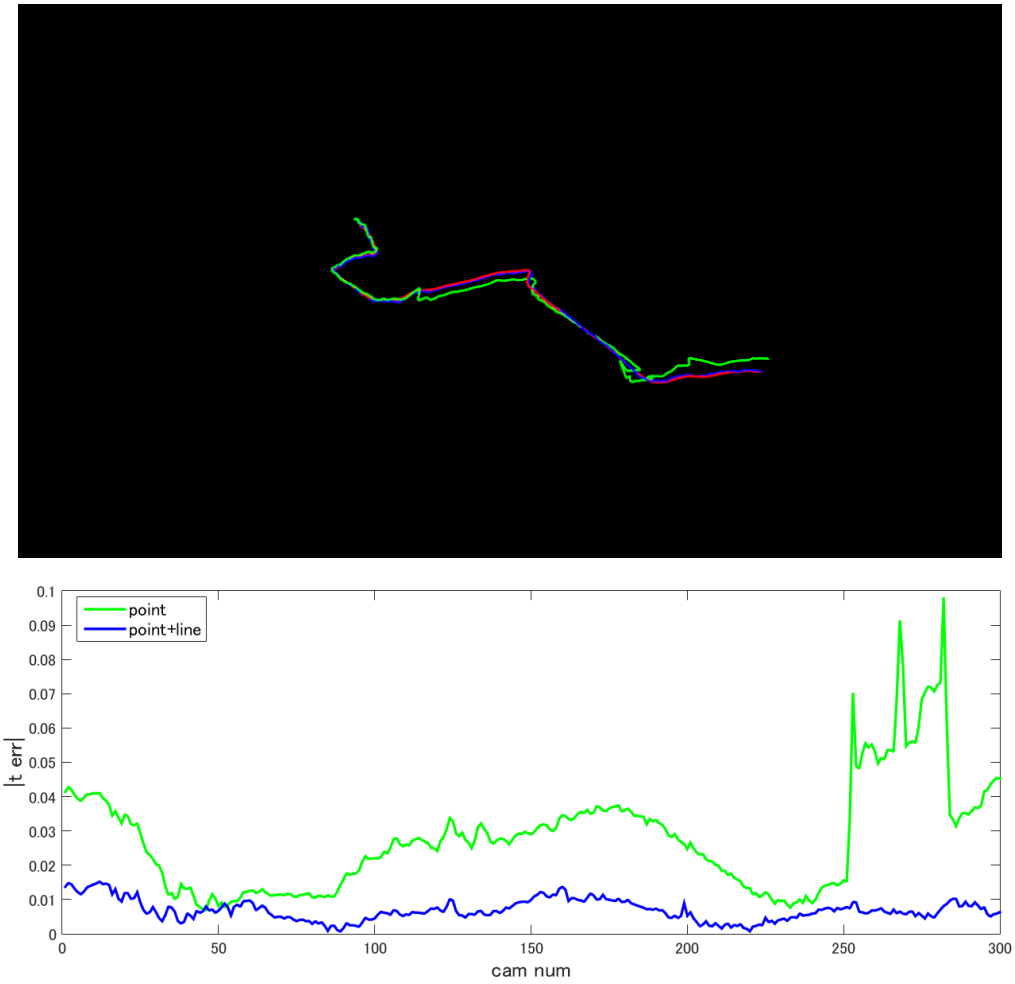


Fig. 4.8: 点モデルと点・直線混合モデルの精度比較．上図:赤線がモーションキャプチャーによって得られた真値の軌跡，緑線が点モデルのみで推定された従来手法，青線が提案手法．下図:従来手法と提案手法におけるカメラ位置の誤差をプロットしたもの．

第5章　結論と今後の展望

5.1　各章のまとめ

本論文では，コンピュータビジョンの根幹技術の一つである SfM に焦点を当て，撮像系の物理モデルを考慮することによってそれを高速・高精度・頑健に計算する手法について述べた．

第 1 章では，SfM の性能劣化の原因となる諸問題を指摘し，それを改善することがコンピュータビジョンにおいて重要であることを述べた．そして，特徴点ベースの方法における SfM の基礎原理について触れ，ローリングシャッターやモーションブラーが特徴点抽出の精度に大きく影響を及ぼし，それに引きづられる形でバンドル調整の精度が落ちることを解説した．加えて，平面追跡に関する最新の研究について触れ，直接法を使ったの平面追跡において，性能劣化の原因が，テンプレート画像と比較する画像の見えの変化，つまり濃淡の不一致によるものであることを確認した．

第 2 章では，特徴点ベースの方法に焦点を当て，ローリングシャッター方式のカメラに対してもバンドル調整可能なように拡張を行なった．RS カメラモデルの作成では，カメラモーションが回転だけしていることを仮定し，アフィンカメラモデルによって近似することで，RS 歪みが 2 つの変形によって表せるものとした．このモデルによって，自己校正の問題と類似した形で CMS の一般表現について導出し，RS 歪みのパラメータのアスペクト成分を固定することによって実用的に重要な CMS の問題を解決した．そして，いくつかの実験を通じて提案した RS SfM の有効性を実証した．

第 3 章では，平面追跡の問題において，追跡対象となる平面が視線に対して大きく傾いたり，遠ざかるなどするとき，撮影画像上での実効的な解像度が低下し，このことが追跡性能を低下させることを指摘した．このような性能低下を防ぐため，画像撮影時の空間方向の標本化過程，および平面追跡の最適化計算で行われる画像の変形をモデル化し，これを考慮した最適化計算を行う方法を示した．核となるアイデアは，追跡する平面パタン（テンプレート）を，追跡中の平面姿勢から作成した線形フィルタを適用することで修正し，これを最適化に用いることである．従来手法に対して計算量の増加は小さく，追跡は実時間で十分行えることを実証した．そして，いくつかの実験を通じて，提案手法は従来

手法よりも明確に優れることを示した．提案手法は，従来手法と比べて平面追跡の限界のより近くに迫れたと考えられる．すなわち，平面と視線が 90 度をなす場合や，平面が無限遠方にある場合を境界とする極限条件に，従来よりも近いところで追跡を行えるようになった．加えて，単一平面追跡から複数平面追跡にどのようにモデルを拡張するかについて解説を行なった．二次元射影変換行列を厳密に定義した式を使い，異なる 2 枚以上の平面を追跡することで，シーンの形状とカメラモーションの同時推定が可能であることを確認した．

第 4 章では，直接法の多平面追跡モデルが一般的な SfM で扱いにくくなっていることを取り上げ，より扱いやすい点モデルの直接法について紹介し，原理について解説した．そして，実環境における実験を通じて，複雑な環境下での SfM にも応用可能であることを確認した．点追跡モデルでは，すべての追跡点が自由度を持ち，空間の形状に関する事前知識をつかうことができないことが問題であった．そのため提案手法では，空間の物理モデルに着目し，点と直線のモデルを融合させることで，同時最適化の枠組みで直線部分の形状を推定できることを示した．

5.2 今後の展望

本研究では，撮像系の物理モデルをカメラ画像の生成過程と物体の形状からくる拘束条件の両面から見ていったが，特徴点ベースの方法と直接法それぞれの場合で，扱える問題の範囲が様々変わっており，これらの垣根を取り払って統一的な SfM の理論を組み立てる必要がある．また，複雑なモデルになるにしたがって多変数のパラメータを推定しなければならず，計算量の制約から未だ実用的な応用に向けての障害も多々存在している．そのため，この物理モデルをいかに精度を保ちつつ簡略な形で表すかが今後の課題となると考えられる．

参考文献

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105–112, Oct. 2011.

[2] Agisoft Inc. Photoscan. http://www.agisoft.com/.

[3] O. Ait-Aider, N. Andreff, J. Lavest, and P. Martinet. Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In *Proc. European Conference on Computer Vision*, 2006.

[4] O. Ait-Aider and F. Berry. Structure and kinematics triangulation with a rolling shutter stereo rig. In *Proc. International Conference on Computer Vision*, 2009.

[5] C. Albl, Z. Kukelova, and T. Pajdla. R6P-Rolling shutter absolute camera pose. In *Proc. Computer Vision and Pattern Recognition*, 2015.

[6] C. Albl, A. Sugimoto, and T. Pajdla. Degeneracies in rolling shutter sfm. In *Proc. European Conference on Computer Vision*, 2016.

[7] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56:221–255, February 2004.

[8] S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 943–948, 2004.

[9] S. Benhimane and E. Malis. Homography-based 2d visual tracking and servoing. *International Journal of Robitics Research*, 26:661–676, July 2007.

[10] D. Caruso, J. Engel, and D. Cremers. Large-scale direct slam for omnidirectional cameras. In *Proc. International Conference on Intelligent Robots and Systems*, September 2015.

[11] R. O. Castle, D. J. Gawley, G. Klein, and D. W. Murray. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *Proc. International Conference on Robotics and Automation, Rome, Italy, April 10-14, 2007*, pages 4102–4107, 2007.

[12] Y. Dai, H. Li, and L. Kneip. Rolling shutter camera relative pose: generalized epipolar geometry. In *Proc. Computer Vision and Pattern Recognition*, 2016.

[13] A. Dame and E. Marchand. Accurate real-time tracking using mutual information. In *Proc. IEEE Internatioanl Symposium on Mixed and Augmented Reality*, pages 47–56, October 2010.

[14] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.

[15] N. Dowson and R. Bowden. A unifying framework for mutual information methods. In *Proc. European Conference Computer Vision*, pages 365–378, May 2006.

[16] N. Dowson and R. Bowden. Mutual information for lucas-kanade tracking (milk): An inverse compositional formulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:180–185, 2008.

[17] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. In *arXiv:1607.02565*, July 2016.

[18] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proc. European Conference on Computer Vision*, September 2014.

[19] C. Geyer, M. Meingast, and S. Sastry. Geometric models of rolling-shutter cameras. In *Proc. OMNIVIS*, 2005.

[20] Google Inc. Ceres Solver. http://ceres-solver.org.

[21] M. Grundmann, V. Kwatra, and D. Castro. Calibration-free rolling shutter removal. In *Proc. International Conference on Computational Photography*, 2012.

[22] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[23] J. Hedborg, P. Forssén, and M. Felsberg. Rolling shutter bundle adjustment. In *Proc. Computer Vision and Pattern Recognition*, 2012.

[24] J. Hedborg, E. Ringaby, and P. Forssén. Structure and motion estimation from rolling shutter video. In *Proc. ICCV Workshops*, 2011.

[25] A. Heyden and K. Åkström. Minimal conditions on intrinsic parameters for euclidean reconstruction. In *Proc. Asian Conference on Computer Vision*, 1998.

[26] A. Heyden and K. Åström. Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In *Proc. Computer Vision and Pattern Recognition*, 1997.

[27] A. Heyden and K. Astrom. Flexible calibration: Minimal cases for auto-calibration. In *Proc. International Conference on Computer Vision*, 1999.

[28] S. Holzer, S. Hinterstoisser, S. Ilic, and N. Navab. Distance transform templates for object detection and pose estimation. In *Proc. Computer Vision and Pattern Recognition*, pages 1177–1184, 2009.

[29] K. Ito, H. Nakajima, K. Kobayashi, T. Aoki, and T. Higuchi. A fingerprint matching algorithm using phaseonly. *IEICE Transactions on Fundamentals*, E87-A(3):682–691, Mar 2004.

[30] C. Jia and B. Evans. Probabilistic 3-D motion estimation for rolling shutter video rectification from visual and inertial measurements. In *Proc. IEEE International Workshop on Multimedia Signal Processing*, 2012.

[31] H. Jin, P. Favaro, and R. Cipolla. Visual tracking in the presence of motion blur. In *Proc. Computer Vision and Pattern Recognition*, pages 18–25, 2005.

[32] H. Jin, P. Favaro, and S. Soatto. A semi-direct approach to structure from motion. *The Visual Computer*, 19(6):377–394, Oct. 2003.

[33] F. Kahl, B. Triggs, and K. Åström. Critical motions for auto-calibration when some intrinsic parameters can vary. *Journal of Mathematical Imaging and Vision*, 13:131–146, 2000.

[34] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2Nd IEEE and ACM International Workshop on Augmented Reality*, IWAR '99, pages 85–, Washington, DC, USA, 1999. IEEE Computer Society.

[35] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for rgb-d cameras. In *Proc. International Conference on Robotics and Automation*, May 2013.

[36] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[37] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1465–1479, September 2006.

[38] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab. A dataset and evaluation methodology for template-based tracking algorithms. In *Proc. IEEE International Symposium on Mixed and Augmented Reality*, pages 145–151, 2009.

[39] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. International joint conference on Artificial intelligence*, pages 674–679, 1981.

[40] L. Magerand, A. Bartoli, O. Ait-Aider, and D. Pizarro. Global optimization of object pose and motion from a single rolling shutter image with automatic 2d-3d matching. In *Proc. European Conference on Computer Vision*, 2012.

[41] E. Malis. Improving vision-based control using efficient second-order minimization techniques. In *Proc. International Conference on Robotics and Automation*, volume 2, pages 1843–1848, April 2004.

[42] S. J. Maybank and O. D. Faugeras. A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8(2):123–151, 1992.

[43] C. Mei, S. Benhimane, E. Malis, and P. Rives. Efficient homography-based tracking and 3-d reconstruction for single-viewpoint sensors. *IEEE Transactions on Robotics*, 24(6), Dec 2008.

[44] C. Mei and I. Reid. Modeling and generating complex motion blur for real-time tracking. In *Proc. Computer Vision and Pattern Recognition*, June 2008.

[45] J. Morel and G.Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2, 2009.

[46] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. International Symposium on Mixed and Augmented Reality*. IEEE, October 2011.

[47] R. A. Newcombe, S. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, editors, *Proc. International Conference on Computer Vision*, pages 2320–2327. IEEE Computer Society, 2011.

[48] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Proc. Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[49] Y. Park, V. Lepetit, and W. Woo. ESM-Blur: Handling and Rendering Blur in 3D Tracking and Augmentation. In *Proc. International Symposium on Mixed and Augmented Reality*, 2009.

[50] M. Pollefeys, R. Koch, and V. L. Gool. Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, pages 7–25, 1999.

[51] E. Ringaby and P. Forssén. Efficient video rectification and stabilisation for cell-phones. *International Journal of Computer Vision*, pages 335–352, 2012.

[52] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. of the 9th European Conference on Computer Vision*, volume 1, pages 430–443, 2006.

[53] O. Saurer, K. Koser, J. Bouguet, and M. Pollefeys. Rolling shutter stereo. In *Proc. International Conference on Computer Vision*, pages 465–472, 2013.

[54] O. Saurer, M. Pollefeys, and G. H. Lee. A minimal solution to the rolling shutter pose estimation problem. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[55] O. Saurer, M. Pollefeys, and G. H. Lee. Sparse to dense 3d reconstruction from rolling shutter images. In *Proc. Computer Vision and Pattern Recognition*, 2016.

[56] F. Schaffalitzky and A. Zisserman. *Multi-view Matching for Unordered Image Sets, or "How Do I Organize My Holiday Snaps?"*, pages 414–431. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[57] G. Silveira and E. Malis. Real-time visual tracking under arbitrary illumination changes. In *Proc. Computer Vision and Pattern Recognition*, pages 1–6, 2007.

[58] G. Silveira, E. Malis, and P. Rives. An efficient direct approach to visual slam. *IEEE Transactions on Robotics*, 24:969 – 979, October 2008.

[59] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press.

[60] F. Steinbruecker, J. Sturm, and D. Cremers. Real-time visual odometry from dense rgb-d images. In *Proc. Workshop on Live Dense Reconstruction with Moving Cameras*, 2011.

[61] C. Strecha, W. von Hansen, and L. Gool. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Proc. Computer Vision and Pattern Recognition*, 2008.

[62] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[63] P. Sturm. Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction. In *Proc. Computer Vision and Pattern Recognition*, 1997.

[64] P. Sturm. Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length. *Image and Vision Computing*, 20:415–426, 2002.

[65] B. Williams, G. Klein, and I. Reid. Real-time SLAM relocalisation. In *Proc. International Conference on Computer Vision*, 2007.

[66] C. Wu. Visual SFM. http://ccwu.me/vsfm/.

[67] C. Wu. Towards linear-time incremental structure from motion. In *Proc. International Conference on 3DVision*, 2013.

[68] C. Wu. Critical configurations for radial distortion self-calibration. In *Proc. Computer Vision and Pattern Recognition*, 2014.

[69] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, November 2000.

付 録A　平面追跡アルゴリズムの GPGPUによる実装

近年，Fig.A.1 の超並列演算装置である GPU(Graphics processing unit) を汎用計算に用いる GPGPU(General-Purpose computing on GPU) の研究が活発化している．画像処理では，ピクセル単位の処理など，並列処理可能な部分が至る所に存在しているため，GPU を有効に使うことで数倍から数百倍のオーダーで計算を高速化できる事例が数多くある．平面追跡の計算も例外ではなく，大規模な処理を並列計算することで，従来よりも計算速度の改善が容易になった．それによって, より緻密な平面追跡の計算モデルを作成できるようになり，ようやく精度・高速性・安定性をすべて満たすようなシステムを開発する下地が整ってきた．

Fig. A.1: NVIDIA 社製の GPU．GPU プログラミングをするための開発環境である CUDA が無償で提供されている．

A.1　GPUのプログラミングの概要

2007年より，NVIDIA社のGPUにおいて，並列プログラムを組むための開発環境であるCUDAが提供されており，汎用計算にGPUを用いる流れが世界中に拡大している．そして現在，CUDAによるGPGPUは，画像処理だけでなく，物理シミュレーションなど，幅広く工・理学分野で利用されている．

GPUで計算することによって，CPUのみで計算した場合よりも，数十倍のオーダーで高速化したという事例はいくつも報告されており，本研究の平面追跡においても，GPUによって高速化が期待できる．しかし，GPUの構造や計算特性を理解しないで単に実装した場合，下手をするとCPUで計算した時よりもはるかに遅くなってしまう可能性がある．この節では，GPUで最適に並列計算をするために必要な知識を整理する．

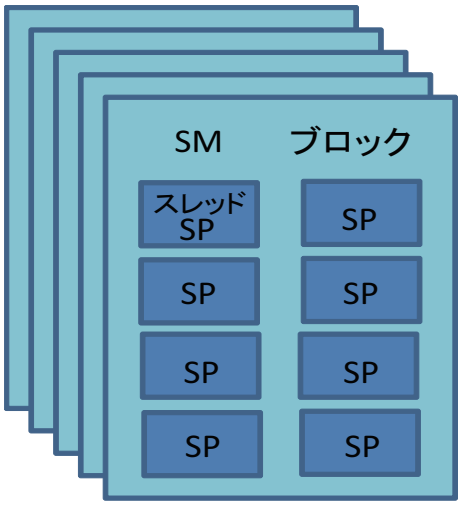


Fig. A.2: GPUのプロセッサ構造について．

GPUでは，並列計算の粒度をスレッド・ブロック・グリッドに分割している．スレッドの集合がブロックであり，ブロックの集合がグリッドである．

GPUのアーキテクチャは現行世代のFermiとそれ以前のものとでは大きく異なっている．そのため，まずは旧世代のG200アーキテクチャについて説明し，補足としてFermi世代に拡張された機能などを紹介する．

GPUのプロセッサの構造はFig.A.2の通りである．コアにあたるものが，SP(Streaming Processor)と呼ばれており，GeForce GTX295では480基搭載されている．並列計算による各々のスレッド処理はここで行われている．CUDAの並列計算の粒度は，スレッド・ブロック・グリッドの順に大きくなっており，スレッドの集合がブロック，ブロックの集

合がグリッドになっている．ブロックの処理は，SP が 8 つ集まってできた SM(Streaming MultiProcessor) が担当している（Fermi アーキテクチャでは 32SP で 1 つの SM を構成）．

SM 内では，SIMD 型のデータ処理が行われており，32 スレッドごとに同じ命令が下される．この同時に処理される 32 個のまとまりは 1 ウォープという単位で呼ばれている．例えば，一つのブロックに 512 スレッド割り当てられていた場合，SM は 16 ウォープの SIMD 命令でブロック内のスレッドを処理している．ウォープの処理内でレジスタの読み書きなどが発生した場合，24 サイクルの遅延が発生するため，SM は実行可能なウォープに切り替えて，常に SP を稼働させるように命令を発行している．

1 つの SM につき保持できるブロック数やスレッド数には上限があり，ブロック単位の処理で使用する共有メモリやスレッドで使用しているレジスタのサイズによって保持できる数が減少していく．保持できるブロックやスレッドの数が減少すると，場合によっては，ウォープの切り替えやメモリの読み込みに時間がかかり，計算効率が低下する恐れがある．

GPU が CPU と大きく異なる点は，メモリバンド幅の広大さと多数のコアの恩恵により，SIMD 型の計算において高スループットを実現していることである．

A.1.1 GPU のメモリの種類と階層構造

ここでは GPU 特有のメモリの種類や特性について述べる．

CPU と GPU で直接扱えるメモリ空間は物理的に異なっている．CUDA プログラミングでは一般的に，CPU 上のメモリについてはホスト側と呼び,GPU 上のメモリについてはデバイス側と呼ぶ．GPU 上で計算を行うには, ホスト側からデータをコピーしなければならない．同様に CPU でデバイス側のデータを扱いたい場合は, ホスト側にコピーしなければならない．

デバイス側のメモリは Fig.A.3 のように階層的な構成を持っている．

Shared Memory

メモリへの書き込みや読み込みは速いが, 他のブロックとデータを共有すること不可であり, 容量も 1 ブロックにつき 16KB と非常に少ない (Fermi アーキテクチャでは L1 キャッシュとメモリを共有している，設定により 16KB か 48KB かを選択できる).

Texture Memory

キャッシュがついているため, 読み込みは高速である．ただし, ホスト側からしか書き込みできない．

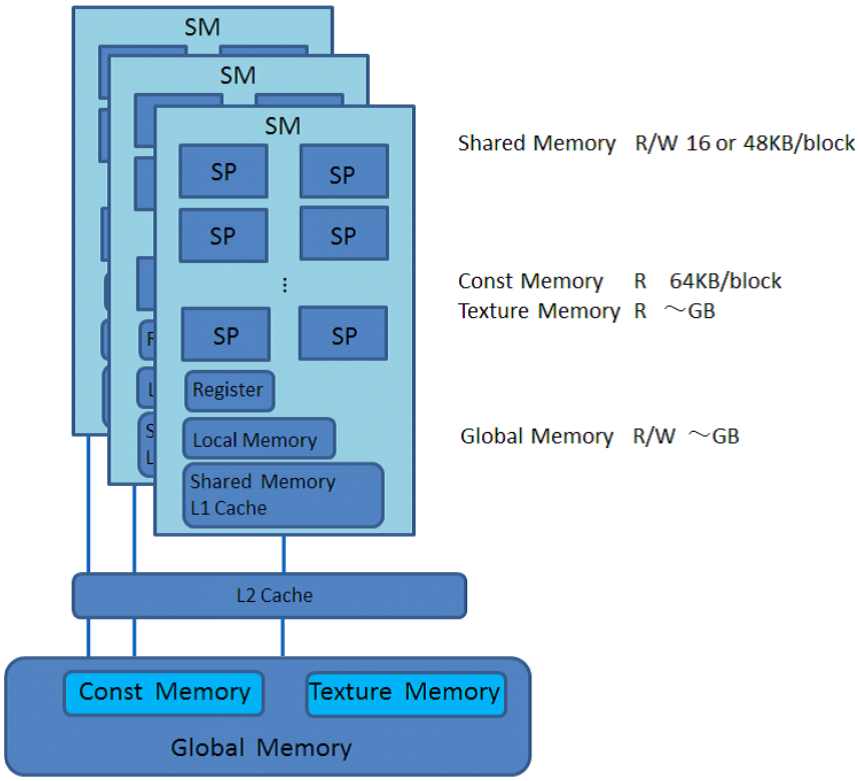


Fig. A.3: GPU のメモリ構造について．

Const Memory

テクスチャメモリと同様にキャッシュがついているため読み込みが高速であるが, 1 つのブロックにつき 64KB の容量のみとなっており, 他のブロックとデータを共有することは不可である．そしてホスト側からしか書き込みできない．

Global Memory

メモリへの書き込みと読み込み共に遅いが, 全てのブロックとデータを共有できる (Fermi アーキテクチャでは L1，L2 キャッシュのおかげでこの問題が緩和されている)

メモリの種類によって容量もアクセス速度も異なるため，計算の内容に応じて適切なメモリを選択しなければならない．

A.1.2 GPU のメモリアクセス

ここでは，効率的な GPU のメモリアクセス方法について述べる．Fig.A.4 のようにメモリ・アクセス命令は半ウォープ 16 スレッドが同時に実行する仕組みになっている．このと

き，32 の倍数から始まる連続したアドレスにあるデータを使うことで，データ移送が一度のメモリアクセスで完了する．しかし，非連続なアドレスのデータをスレッドで計算する際, 何度か分けてデータを読み込む必要が出てしまう．このような場合，読み込みに数百サイクルかかるグローバルメモリに複数回アクセスするため，速度が極端に低下する．グローバルメモリへのランダムアクセスを避け，上記の条件を満たすように連続したアドレスの情報を読み込むことを Coalesced Access と呼び, 最も効率の良いメモリアクセス方法となっている．

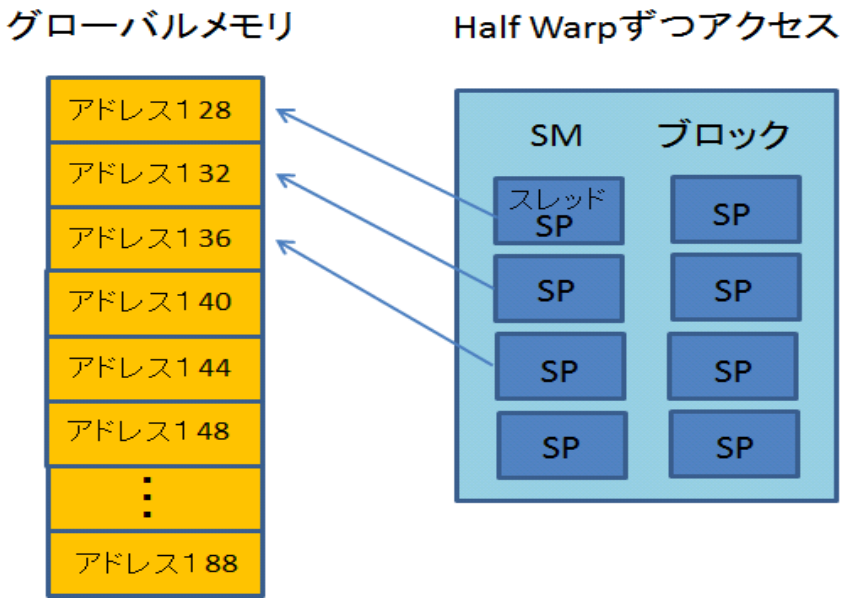


Fig. A.4: グローバルメモリへの Coalesced Access について．

シェアードメモリに対しても，Coalesced Access のように効率的なメモリアクセス方法が存在している．Fig.A.5 のように，シェアードメモリは１６個のバンクによって構成されており，グローバルと同様に半ウォープずつアクセスされる．しかし，スレッド間で同じバンクに属するデータへ同時にアクセスする場合，重複した分だけメモリアクセスにかかる時間も増加する．例えば，アドレスを一つ飛ばしでアクセスする場合，同じバンクのデータを二回参照することになるため，メモリアクセスも二倍の時間がかかる．これを 2-way バンクコンフリクトと呼ぶ．したがって，シェアードメモリを使って計算を最適化する場合は，バンクの重複に気を付けるべきである．

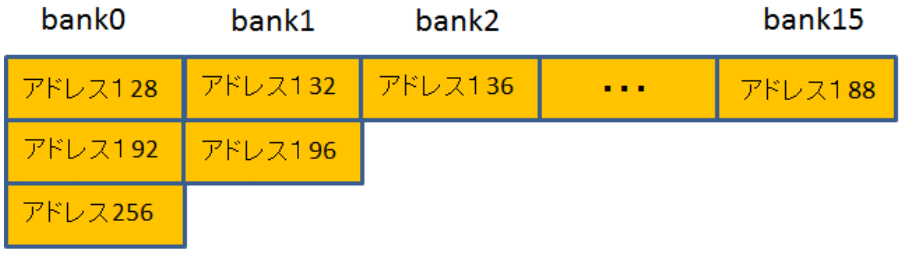


Fig. A.5: シェアードメモリのバンクコンフリクト．異なるバンクは同時にアクセス可能

A.2　GPU を用いた ESM の実装

ここでは, 具体的に ESM を GPU でどのように実装するかを述べる．ESM は 3 章でも説明したように画像の微分や差分など並列化が容易な計算が多いが，Eq.(1.55) のように画素数 ×8 の大きな連立方程式の求解では，メモリの使い方に注意しながら並列化しなければいけない．また,ESM にはテンプレート画像の情報から事前に用意できるデータも存在する為，それらのデータの扱いかたについても述べる．

A.2.1　テンプレート画像を使った事前処理

ここでは，トラッキングの事前に準備できる計算をどのように扱うか述べる．必要な計算は，テンプレート画像の輝度値を空間微分した行列 $\mathbf{J}_{I^*}$ を求めること，テンプレート画像の位置情報を持った行列とリー代数の基底が格納された定数行列の積 $\mathbf{J}_w\mathbf{J}_\mathbf{H}$ を求めることの主に 2 点である．ここで計算された行列はトラッキングの最中に何度も読み込まれるため，適切な GPU メモリを選択する必要がある．

まずは，Eq.(1.54) にあるテンプレート画像の空間微分 $\mathbf{J}_{I^*}$ の扱いについて述べる．

Fig.A.6 のようにして得られたテンプレート画像の i 行 j 列のピクセルにおける輝度値を x 方向, y 方向に微分すると以下の式で表される．

$$I_x(i,j) = I_x(i,j+1) - I_x(i,j) \tag{A.1}$$

$$I_y(i,j) = I_x(i+1,j) - I_x(i,j) \tag{A.2}$$

これは x 方向の微分値と y 方向の微分値がそれぞれ画像形式のデータになっているため，それぞれをテクスチャメモリに格納することで後の計算で高速に読み出すことができる．

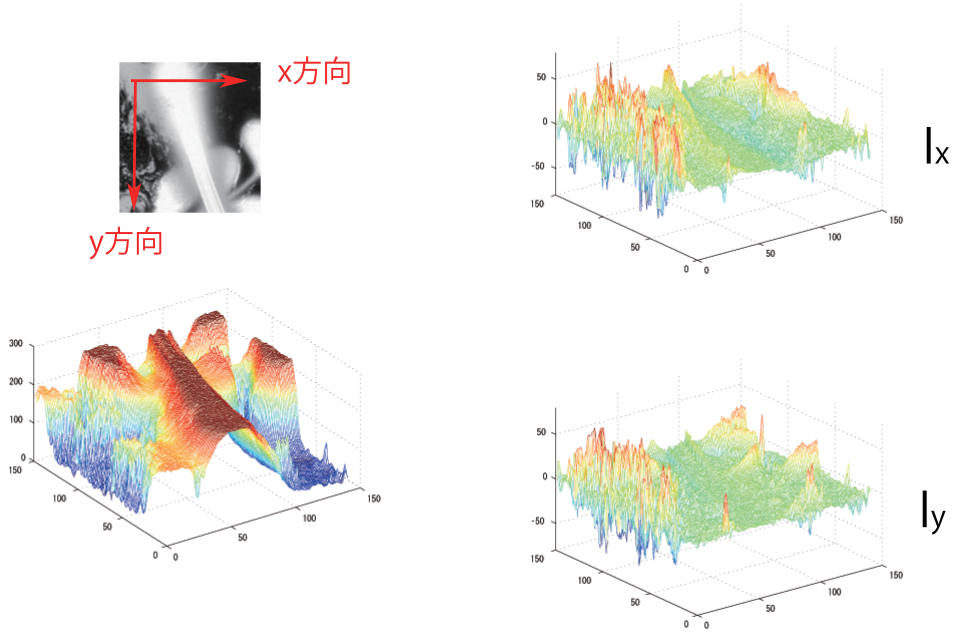


Fig. A.6: テンプレート画像の微分．

次に，テンプレート画像のサイズに依存する定数行列 $\mathbf{J}_w\mathbf{J}_\mathbf{H}$ の扱いについて述べる．テンプレート画像の i 番目のピクセルの座標を $\mathbf{p}_i^* = [u_i^*, v_i^*, 1]^T$ とすると $\mathbf{J}_w$ は以下のように定義される．

$$\mathbf{J}_w = \begin{bmatrix} \mathbf{p}_i^{*T} & \mathbf{0} & -u_i^*\mathbf{p}_i^{*T} \\ 0 & \mathbf{p}_i^{*T} & -v_i^*\mathbf{p}_i^{*T} \end{bmatrix} \tag{A.3}$$

Eq.(A.3) は 2×9 の行列であり，テンプレートの画素数分存在している．また，$\mathbf{J}_\mathbf{H}$ は 9×8 の定数行列であり，以下のように定義される．

$$\mathbf{J}_\mathbf{H} = \begin{bmatrix} [\mathbf{A}_1]_\mathbf{v} & [\mathbf{A}_2]_\mathbf{v} & \cdots & [\mathbf{A}_8]_\mathbf{v} \end{bmatrix} \tag{A.4}$$

ここで $[\mathbf{A}_i]_\mathbf{v}$ は，リー代数の基底を行ごとに読み込んでベクトルに直したものである．$\mathbf{J}_w\mathbf{J}_\mathbf{H}$ を計算することで 1 つのピクセル毎に 2×8 の行列が生成されるが，データをそのままグローバルメモリに格納した場合，トラッキング中の計算でデータの読み込みに時間がかかる．そのため行列を成分ごとに分割し，Fig.A.7 のように $\mathbf{J}_w\mathbf{J}_\mathbf{H}(1,1)$ 成分の画素数分のデータ，$\mathbf{J}_w\mathbf{J}_\mathbf{H}(1,2)$ 成分の画素数分のデータという形で，16 枚の画像としてテクスチャメモリに

格納している．Fermi 世代の GPU ではグローバルメモリにもキャッシュがついたため，テクスチャメモリにこだわる必要はなくなった．

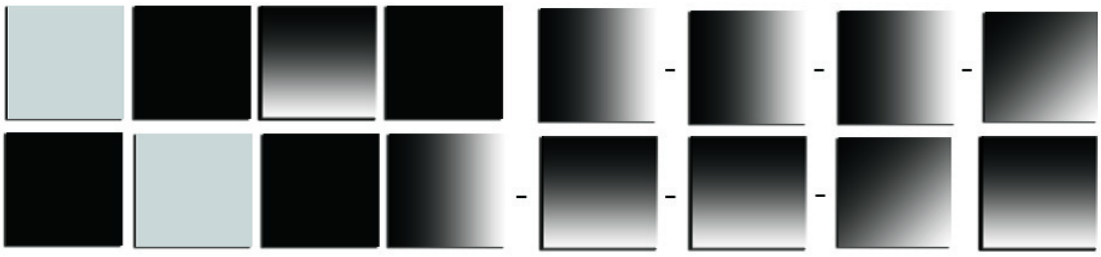


Fig. A.7: $\mathbf{J}_w\mathbf{J}_\mathbf{H}$ を 16 枚の画像としてテクスチャメモリへ格納．

A.2.2　追跡中の処理

ここでは，トラッキングの計算速度にもっとも関わる反復処理の方法について説明する．必要な手順は，更新されたホモグラフィによって画像を変換し現在の推定領域の画像 I を手に入れ，空間微分画像 $\mathbf{J}_I$ と時間微分画像 $\mathbf{y}(\mathbf{0})$ を求める．その後，Eq.(1.54) のように $\mathbf{J}_{esm}$ を計算する．最終的に $\mathbf{J}_{esm}x = -\mathbf{y}(\mathbf{0})$ の優決定の連立方程式を解き，ホモグラフィの更新に必要なパラメータ $\mathbf{x}$ を導出する．まず，一つ目のホモグラフィを用いて追跡領域の画像を GPU で計算する方法について述べる．

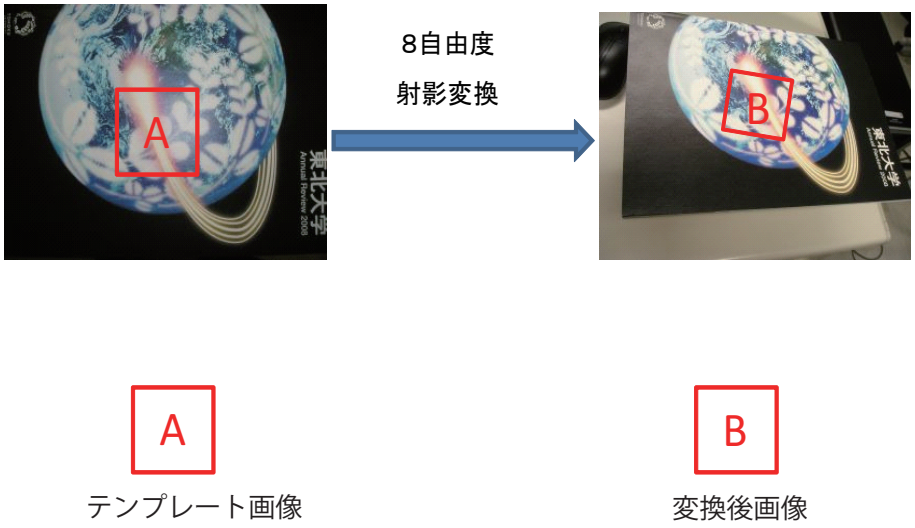


Fig. A.8: ホモグラフィによる平面パターンの変形．

Fig.A.8 のように，テンプレート領域の座標を Eq.(1.24) を用いて座標変換すると変換後の領域が得られる．ホモグラフィによって変換されたピクセルの座標は整数型でないため，ピクセルの濃淡値を補間して得る必要がある．GPU のテクスチャメモリには，線形補間の機能がついているため，高速に座標変換された後のピクセル値を計算することができる．

推定領域の空間微分 $\mathbf{J}_I$ については, テクスチャメモリに格納された画像情報を 1 ピクセルずらしてからホモグラフィ変換したものと変換後画像との差を取ることで求めている．$\mathbf{y(0)}$ は，テクスチャメモリに記録したテンプレート画像と変換後画像の差分をとって並列計算している．次に，$\mathbf{J}_{esm}$ を求めるために $\frac{1}{2}(\mathbf{J}_{I^*} + \mathbf{J}_I)\mathbf{J}_w\mathbf{J_H}$ を計算する．前節で $\mathbf{J}_w\mathbf{J_H}$ を 16 枚の画像としてテクスチャメモリに保存したため，$\mathbf{J}_{esm}$ の i 行 j 列の成分を以下のように，ベクトルの成分同士の掛け合わせとして高速に計算することができる．

$$\sum_{i=1}^{\text{画素数}} \sum_{j=1}^{8} \mathbf{J}_{esm}(i,j) = \frac{1}{2}(\mathbf{J}_{I^*}(i) + \mathbf{J}_I(i))_x \mathbf{J}_w \mathbf{J}_{\mathbf{H}1j}(i) + \frac{1}{2}(\mathbf{J}_{I^*}(i) + \mathbf{J}_I(i))_y \mathbf{J}_w \mathbf{J}_{\mathbf{H}2j}(i) \tag{A.5}$$

我々は様々な方法を用いて優決定の連立方程式 $\mathbf{J}_{esm}\mathbf{x} = -\mathbf{y(0)}$ を解いたが，ここでは GPU を用いて最も速く計算できた提案手法についてのみ述べる．

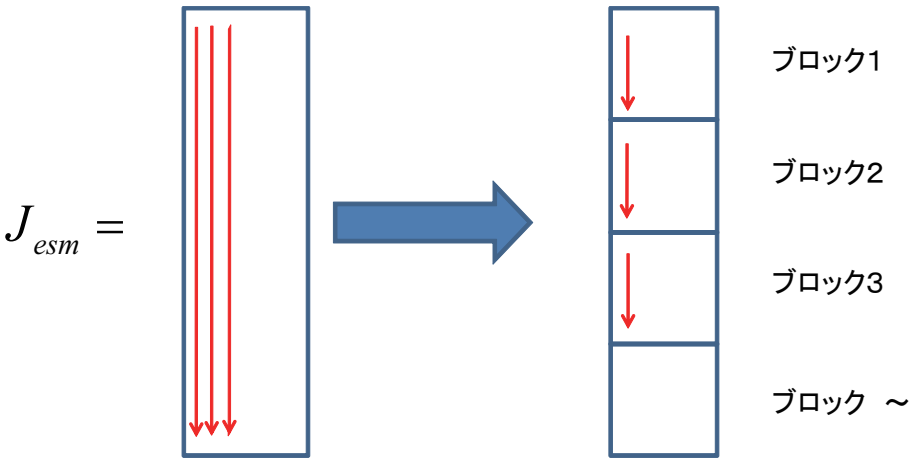


Fig. A.9: 効率的なメモリ分割.

我々の提案手法では，最小二乗解を求める優決定の連立方程式を同じく最小二乗解を求める正規方程式に変形する．その後，ホスト側に正規方程式を転送し，Cholesky 分解を用いて解を求める．正規方程式に直すには Eq.(A.6) のように, 連立方程式の両辺に左から

$\mathbf{J}_{esm}^T$ を掛ければ良い．

$$\mathbf{J}_{esm}^T\mathbf{J}_{esm}\mathbf{x} = -\mathbf{J}_{esm}^T\mathbf{y}(\mathbf{0}) \tag{A.6}$$

ここで，今まで計算してきた行列 $\mathbf{J}_{esm}$ は Fig.A.9 にあるように縦の順でグローバルメモリに格納されている．これをそのまま読み込んで転置行列を掛けた場合，グローバルメモリは読み込みが遅いため処理速度も低下する．それを解決するためにグローバルメモリのデータをメモリへの読み込みが高速なシェアードメモリに転送し，$\mathbf{J}_{esm}^T\mathbf{J}_{esm}$ を計算することにした．ただしシェアードメモリは 1 ブロックあたりサイズが 16KB しかない為，行列を小行列に分解してから計算している．もう一つ注意しなければならないのは，グローバルメモリからシェアードメモリにデータを転送する際，上記で述べた Coalesced Access を守ることである．したがって，グローバルメモリに格納されている順でシェアードメモリに格納していくことが重要となる．正規方程式の導出は，Fig.A.10 のようにブロック同士の積を足し合わせた形となり，並列計算が可能となる．以上によって求められた 8×8 の正規方程式は対称行列となるため，その特性を生かし Cholesky 分解で求解を行った．

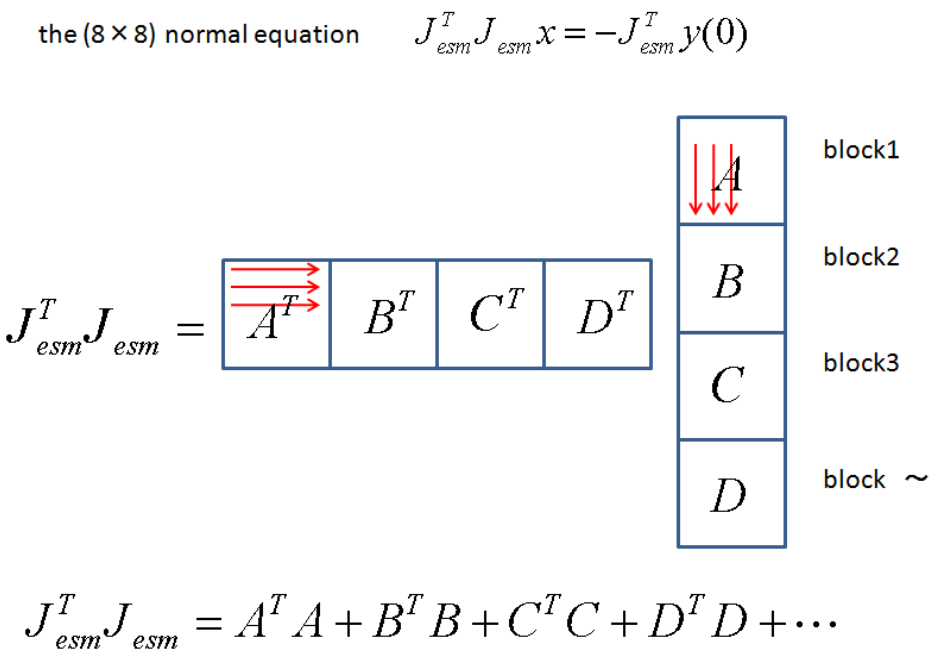


Fig. A.10: Shared Memory を用いた $\mathbf{J}_{esm}^T \mathbf{J}_{esm}$ の計算.

以下に GPU プログラムの核心部分である，正規方程式の計算を記す.

```
//ホスト側からカーネル関数を実行するときに用いるスレッド数
#define BLKHIEIGHT (256)

//カーネル関数を実行するときのブロック数
#define BLKSIZE  (テンプレートのピクセル数/BLKHEIGHT)

//シェアードメモリで計算する部分行列のサイズ
#define SUBMAT_SIZE (8*9)

//================================================================
//グローバルメモリ(ab)から[Jesm|y(0)]を読み込み，シェアードメモリ(sub_ab)に格納.
//その後，部分行列の掛け合わせTranspose(Jesm)*sub_abを実行し
//グローバルメモリ(sub_abs)に実行結果を格納する
//================================================================
```

```
__global__ void
calcJtJy_kernel( int nrows, float *ab, float *sub_abs )
{
    //シェアードメモリの宣言
    __shared__ float sub_ab[BLKHEIGHT][9];

    //グローバルメモリからコアレッシングにデータを
    for (int i = 0; i < 9; i++) {
        sub_ab[threadIdx.x][i]=
            ab[i*nrows+blockIdx.x*BLKHEIGHT+threadIdx.x];
    }

    //シェアードメモリに書き終わるまでスレッドを同期させる
    __syncthreads();

    int row = threadIdx.x%8;
    int col = threadIdx.x/8;

    //(8*BLKHEIGHT)行列と(BLKHEIGHT*9)行列の積を取り
    //グローバルメモリ格納
    if (col < 9) {
        float val = 0.;
        for (int el = 0; el<BLKHEIGHT; el++)
            val += sub_ab[el][row]*sub_ab[el][col];

        sub_abs[blockIdx.x*SUBMAT_SIZE+col*8+row] = val;
    }
}

//============================================================
//分割した部分行列をすべて足し合わせる操作
//============================================================
__global__ void
```

```
sumJtJy_kernel( int nblks, float *sub_abs, float *res )
{
    float sum = 0.0;

    for (int i = 0; i<BLKSIZE; i++)
        sum += sub_abs[i*SUBMAT_SIZE+blockIdx.x*8+threadIdx.x];

    res[blockIdx.x*8+threadIdx.x] = sum;
}
```